ID Verifier

Apresentação do Projeto e Teste

Apresentando uma proposta de solução na pré compra e pós compra para BASF, utilizando de Cybersecurity, foi desenvolvido um modelo para a verificação de ID's que estão relacionados ilicitamente dentro da empresa.

O ID Verifier conta com uma Inteligência Artificial, que realiza a busca de um certo ID digitado, em bases de dados, e retorna com uma resposta de acordo com:

- As letras digitadas
- Os números digitados
- Uma busca nas bases de dados:
- Base de Furtos
- Base de Fraudes
- Base de Falecimentos
- Base de Compras Canceladas

Foram realizados testes com as bases de dados sintéticas e o modelo concluiu com êxito todas as tarefas, conforme vídeo abaixo:

https://drive.google.com/file/d/1fxzsDXIGNcb8gKCcQ9R__we92mgz6ry4/view?usp=sharing

2º Desenvolvimento do modelo

Para o desenvolvimento deste modelo, utilizamos uma serie de bibliotecas do Python, dentre elas:

- Streamlit - Para prototipagem do modelo via WEB
- Ngrok - Como tunel para a prototipagem
- Requests - Para puxar as bases de dados dos ID's proibidos
- Json - Para utilizar as bases de dados e percorrer os arquivos.
- Regex (Regular Expressions) - Para realizar a verificação dos ID's digitados
- Fuzzywuzzy - como Inteligencia Artificial, em conjunto com Regex para realizar a busca destes ID's nas bases de dados.

Codigos:

Instalando os recursos necessários. (Streamlit/ace, Ngrok, Fuzzy)

```
!pip install -q streamlit
!pip install -q streamlit-ace
!pip install -q pyngrok
!pip install -q fuzzywuzzy[speedup]
```

Criando o aplicativo com streamlit.

```
!streamlit run app.py &>/dev/null&
```

Instalando Ngrok separadamente. (No meu caso, o colab estava reclamando que o ngrok estava sendo importado dentro da aplicação principal, uma maneira de resolver isso que achei, foi importa-lo separadamente).

```
from pyngrok import ngrok
```

Aplicação. Importando bibliotecas e escrevendo no arquivo do aplicativo criado anteriormente.

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

Definindo as bases de dados.

```
#set up data sets
dataSet_idFurto = r.get("https://henricobela.github.io/data/json
/idfurto.json")
dataSet_idFraudes = r.get("https://henricobela.github.io/data/json
/idfraude.json")
dataSet_idFalecidos = r.get("https://henricobela.github.io/data/json
/idfalecido.json")
dataSet_idCancelCompra = r.get("https://henricobela.github.io/data/json
/idcancel.json")
```

Definindo as funções que irão fazer as verificações dos ID's digitados.

```
#functions to verify
def idFurtoVerifier(id):
  idS = re.search("[\D]{2}", id.lower())
  number = re.search("(?i)(\d+)", id.lower())
  idString = str(idS[0])

  if idString and number:
    ids = dataSet_idFurto.json()[idString]
    for i in ids:
      for value in i.values():
        if str(id) == str(idString) + str(value):
          return "deny reason: furto"
  return "allow reason: not furto"


def idFraudeVerifier(id):
  idS = re.search("[\D]{2}", id.lower())
  number = re.search("(?i)(\d+)", id.lower())
  idString = str(idS[0])

  if idString and number:
    ids = dataSet_idFraudes.json()[idString]
    for i in ids:
      for value in i.values():
```

```
        if str(id) == str(idString) + str(value):
          return "deny reason: fraude"
  return "allow reason: not fraude"


def idFalecidoVerifier(id):
  idS = re.search("[\D]{2}", id.lower())
  number = re.search("(?i)(\d+)", id.lower())
  idString = str(idS[0])

  if idString and number:
    ids = dataSet_idFalecidos.json()[idString]
    for i in ids:
      for value in i.values():
        if str(id) == str(idString) + str(value):
          return "deny reason: falecido"
  return "allow reason: not falecido"


def idCancelVerifier(id):
  idS = re.search("[\D]{2}", id.lower())
  number = re.search("(?i)(\d+)", id.lower())
  idString = str(idS[0])

  if idString and number:
    ids = dataSet_idCancelCompra.json()[idString]
    for i in ids:
      for value in i.values():
        if str(id) == str(idString) + str(value):
          return "deny reason: cancelCompra"
  return "allow reason: not cancelCompra"
```

Definindo a função de decisão. (Responsável por verificar se é um id proibido e retornar uma resposta)

```
def idVerifier(id):
  verifierFurto = idFurtoVerifier(id)
  verifierFraude = idFraudeVerifier(id)
  verifierFalecido = idFalecidoVerifier(id)
  verifierCancel = idCancelVerifier(id)

  if verifierFurto == "deny reason: furto":
    return st.error("Voce não pode realizar a compra!\nRazão: ID consta
na base de Furtos")

  elif verifierFraude == "deny reason: fraude":
    return st.error("Voce não pode realizar a compra!\nRazão: ID consta
na base de Fraudes")

  elif verifierFalecido == "deny reason: falecido":
    return st.error("Voce não pode realizar a compra!\nRazão: ID consta
na base de Falecidos")

  elif verifierCancel == "deny reason: cancelCompra":
    return st.error("Voce não pode realizar a compra!\nRazão: ID consta
na base de Compras Canceladas")


  return st.success("Voce pode realizar a compra!")
```

Definindo a função principal. (Para criação do frontend com streamlit)

```
#function main
def main():
  #page config
  st.set_page_config(
      page_title="ID Verifier",
      page_icon="",
      layout="centered",
      initial_sidebar_state="expanded"
      #, menu_items={
      #     'Get Help': 'https://henrico.atlassian.net/l/c/yPzAHC21',
      #     'Report a bug': "https://www.linkedin.com/in/henricobela/",
      #     'About': "https://github.com/henricobela/idVerifier"
      #     }
          )



  #components
  st.title(" ID Verifier ")
  st.subheader("Digite seu ID abaixo: EX:(id12345)")



  #form
  form = st.form(key="form")
  with form:
    idNumber = st.text_input("ID:")
    submit = st.form_submit_button(label="SEND")



  #verification
  if submit:
    if not idNumber:
      st.warning("Por favor digite seu ID!!!")
    else:
      try:
        idVerifier(idNumber)
      except:
        st.warning("Por favor, digite o ID corretamente conforme
exemplo: EX: id12345")
```

Chamada final.

```
#call main
if __name__ == "__main__":
  main()
```

Execução do app.

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

3º Decisão final e testes de metricas

O modelo reagiu bem apenas com regex, portanto não foi necessário a utilização do Fuzzy, porém, testes realizados com o mesmo também geraram bons resultados. Mas para este tipo de solução, achamos melhor utilizar apenas o regex.

Para a decisão de melhor métrica, foi concluído que a partir dos seguintes testes, todas as metricas tiveram um bom resultado.

#	id	id probido?	O algoritmo classificou como id proibido
1	7520	Sim	Sim
2	75205	Não	Não
3	9141	Sim	Sim
4	7603	Sim	Sim
5	7527	Sim	Sim
6	75278	Não	Não
7	8099	Sim	Sim
8	80998	Não	Não
9	4369	Sim	Sim
10	2940	Sim	Sim

id proibido	não id proibido mas é proibido
Verdadeiro Positivo	Falso Positivo
7	0
Verdadeiro Negativo	Falso Negativo
3	0
não proibido	não proibido mas não é id proibido

Precisão	F1-Score
1	1
Acurácia	Recall
1	1

Concluímos que a melhor métrica a ser utilizada é a Precisão, pois com o Precisão podemos avaliar exatamente quais os ID’s que são proibidos, e quais não são.

Por exemplo, ao classificar uma ação como um bom investimento, é necessário que o modelo esteja correto, mesmo que acabe classificando bons investimentos como maus investimentos (situação de Falso Negativo) no processo. Ou seja, o modelo deve ser preciso em suas classificações, pois a partir do momento que consideramos um investimento bom quando na verdade ele não é, uma grande perda de dinheiro pode acontecer.