```
# funcoesTCC.py
#
# !/usr/bin/env python
# coding: utf-8

# # Classificação dos modelos de motocicleta a partir da descrição do produto

import pandas as pd, numpy as np
import re
import pickle

# ### Importa as stopwords da língua portuguesa
# Importar lista de Stopwords
from nltk.corpus import stopwords
stopwords = set(stopwords.words('portuguese'))
# Palavras a adicionar na lista de stopwords estão contidas em um arquivo csv externo
dfsw = pd.read_csv('./bases/stopwords.csv', encoding='ISO-8859-1')
stopwords_df=sorted(list(dfsw['stopword']))
# Atualizar stopwords
stopwords.update(stopwords_df)

# ### Carrega lista de aplicações e cria palavrasChave
# carrega a lista de marcas de motos do arquivo
dfaplicacoes=pd.read_csv('./bases/Aplicacoes.csv')
# remove caracteres especiais ou soltos e termos duplicados, salvando na lista
palavrasChave=sorted(set(re.sub(r"\b \w \b",
                                ' ',
                                re.sub(r"[/<>()|\+\$%&#@\'\"]+",
                                       " ",
                                       "
".join(dfaplicacoes['APLICACOES'].tolist()))).split()))

# #### Função de limpeza de dados irrelevantes para a classificação e remoção de stopwords
def limpaDescricao(descricao): #
    descricao=descricao.lower() #transformar em minúsculas
    # remove top (1045) e variantes
    descricao=re.sub(r'\b[ (-]*top \( *1045 *\)[-) ]*\b',' ',descricao)
    # remove códigos numéricos entre parênteses com -*/
    descricao=re.sub(r"\( *\d*[\/\*\-\d]*\d* *\)", ' ', descricao)
    # remove a ocorrência de "código e etc." e o termo seguinte começado com número
    # att: (alguns tem hífen ou asterisco) (colocar antes de remover pontuação)
    descricao=re.sub(r"\b(invoice|código|codigo|cod|cód|(certificado|cert)(no|nr|)|ref)[0-9a-z/\-\*\.\:]* *\d[^ ]+", ' ', descricao)
    # remove identificação de referência de engrenagens dos kits (antes da pontuação)
    #descricao=re.sub(r"([^a-z]|)(ho|uo|h|l|t|ktd|sm|m|d| x|elos )\d{1,}[ x\-\/,);.]|[ x\-\/(]*\d{1,}(ho|uo|h|l|z|t|ktd|m|d|x| dentes| elos)[ \-\/,;)]", ' ', descricao) # 00h
    descricao=re.sub(r"\d*(ho|uo|h|l|t|ktd|sm|m|d|elos )\d{1,}[ \-\/,);.]|[ \-\/(]*\d{1,}(ho|uo|h|l|z|t|ktd|m|d| dentes| elos)", ' ', descricao) # 00h
    # substitui os termos "s/re" e "s/ret" por "sem retentor"
    descricao=re.sub(r"\b(s\/re|s\/ret)\b", 'sem retentor', descricao)
    # substitui os termos "c/re" ou "c/ret" por "com retentor"
    descricao=re.sub(r"\b(c\/re|c\/ret)\b", 'com retentor', descricao)
    # substitui o termo "aplicação" e "modelo" emendado com outro
```

```
    descricao=re.sub(r"aplicacao", "aplicacao ", descricao)
    descricao=re.sub(r"modelo", "modelo ", descricao)
    # remove códigos no início da descrição
    descricao=re.sub(r"^\b\d{2,}[^ ]*\b", ' ', descricao)
    descricao=re.sub(r"^k[^ ]+", ' ', descricao)
    descricao=re.sub(r"- | -|[\\\+,.:;!?/]+", ' ', descricao) #remover pontuação
(att: "- " ou " -")
    #correção de erros de digitação comuns
    termos={'titan': ['titian','tita','tintan','tit'],
            'honda': ['hond','hnda','hon'],
            'twister': ['twist', 'twiste'],
            'dafra kansas': ['dafra kan'],
            'tenere': ['tener','tenerre'],
            'broz': ['bros','bross'],
            'titan 150': ['titan150'],
            'broz 150':
['bross125.','bros125.','broz125','bross150.','bros150.','broz150'],
            'pop 100':['pop100'],
            'phoenix':['phoeni','phenix'],
            'c100': ['c 100']}
    for termo in termos:
        for termoerrado in termos[termo]:
            descricao=re.sub(r"\b"+termoerrado+r"\b", termo, descricao)
    descricao=re.sub(r"[/<>()|\+\\$%&#@\'\"]+", ' ', descricao) #remover
carcteres especiais
    # remove a ocorrência de medidas tipo 00x000x00 ou 000x0000
    descricao=re.sub(r"\b\d{1,}(x|\*)\d{1,}(x|\*)\d{1,}|\d{1,}(x|\*)\d{1,}\b", '
', descricao)
    # remove identificação de quantidades, unidades, peças e conjuntos
    descricao=re.sub(r"\b(\d* *(conj|und|uni|pc|pç|pec|peç)( \w|\w)+?)\b", ' ',
descricao)
    # remove identificação de mais de 4 dígitos com ou sem letras no início e no
final
    descricao=re.sub(r"\w+\d{4,}\w+", ' ', descricao)
    # remove números de 4 dígitos ou mais começados de 2 a 9
    descricao=re.sub(r"\b[02-9]\d{3,}\b", ' ', descricao)
    # remove identificação de termos começados por zero
    descricao=re.sub(r"\b0\d*\w+?(?=\b)", ' ', descricao)
    # remove a ocorrência de "marca " e o termo na lista até o próximo espaço
    for marca in ['kmc *gold','am *gold','king','bravo *racing','riffel *top']:
        descricao=re.sub(r"\bmarca[ :\./]*"+str(marca)+r"[^ ]*", ' ', descricao)
# colocar antes das stopwords
    descricao=re.sub(r"marca[ :\./]*\w+", ' ', descricao)
    descricao=re.sub(r"(^-| -|- )", ' ', descricao)
    # remove stopwords mantendo a ordem original da descrição
    descricao=list(dict.fromkeys(descricao.split())) # cria lista com termos
únicos
    descricao=" ".join([x for x in descricao if x not in set(stopwords)]) #
exclui stopwords
    # limpa os número que não estão na lista de aplicações (colocar depois das
stopwords)
    desc=descricao.upper().split() # quebra a descrição
    dif=list(set(descricao.upper().split()).difference(palavrasChave)) # pega os
termos diferentes de palavrasChave
    [desc.remove(x) for x in desc if (x in dif and x.isnumeric())] # exclui de
desc os termos numéricos diferentes
    descricao=" ".join(desc).lower() # volta para texto
    #remover hífen, letras ou números soltos (deixar duplicado mesmo)
    descricao=re.sub(r"(^-| -|- |\b\w\b)", ' ', descricao)
```

```
    descricao=re.sub(r"(^-| -|- |\b\w\b)", ' ', descricao)
    #substitui remove o i das cilindradas: ex.: 125i por 125
    termos=re.findall(r"\d{1,}i\b",descricao)
    if termos:
        for termo in termos:descricao=descricao.replace(termo,termo[:-1])
    # remove espaços em excesso (colocar no final)
    descricao=re.sub(r" {2,}", ' ', descricao)
    descricao=descricao.strip()
    # retorna a descricao como saída da função
    return descricao # retorna a descrição

def achaPalavraChave(descricao):
    palavras=[]
    descricao=descricao.upper()
    desc=descricao.split()
    for palavra in palavrasChave:
        if palavra in desc:
            palavras.append(palavra)
        else:
            if palavra.isnumeric():
                pat=r"[0-9]*"+str(palavra)+r"[0-9]*"
            elif palavra.isalpha():
                pat=r"[A-Z]*"+str(palavra)+r"[A-Z]*"
            else:
                pat=r"\b"+palavra+r"\b"
            a = re.findall(pat,descricao)
            if len(a)>0:
                 # adiciona resultado nas palavras se o resultado estiver em palavrasChave
                palavras+=[a[i] for i in range(len(a)) if a[i] in palavrasChave]
    palavras=list(set(palavras)) # remove duplicados
    palavras=" ".join(palavras) # converte para string
    return palavras.lower()

# termos que iniciam item da descrição correspondem a marca
# As que começam com espaço devem permanecer assim, pois há outros modelos com o mesmo final
Marcas = {'HONDA': ['CG','CD','CBX','CB','CBR','CRF','BIZ','BROS','BROZ','XL','FAN','XR','XRE'
                    'DREAM','TITAN','TODAY','TWIN','POP','NX','NXR','TWISTER','HORNET',
'AMERICA','BOLDOR','DUTY','FIREBLADE','FURY','WING','LEAD','MAGNA','NL',
                    ' NC','NSR','NC','NXR','PACIFIC','COAST','SHADOW','STRADA','STUNNER','HAWK',
'SUPERBLACKBIRD','TORNADO','TURUNA','XRV','AFRICA','VALKYRIE','VARADERO',
                    'VFR','VLR','VTR','VTX','TRANSALP'],
          'YAMAHA': ['AEROX','ALBA','AXIS','BWS','DRAG ','DT','FZ','FJ','RD','TENERE',
'MT','XF','XJ','XS','XT','XZ','YF','YZ','LANDER','GLADIATOR','GRIZZLY',
                    'YBR','YZ','VIRAGO','FACTOR','EC','CRYPTON','FAZER','JOG','LANDER',
'FROG','LIBERO','MAJESTY','MEST','MIDNIGHT','MORPH','NEO','PASSOL'],
          'DAFRA':
['APACHE','CITYCOM','KANSAS','LASER','NEXT','RIVA','ROADWIN','ZIG','SPEED'],
          'SUZUKI': ['KATANA','YES','INTRUDER'],
```

```
          'ZONGSHEN': ['ZS'],
          'KASINSKI': ['COMET','MIRAGE'],
          'POLARIS': ['SPORTSMAN','RZR','RANGER'],
          'KAWASAKI':
['NINJA','VERSYS','VOYAGER','GTR','KDX','KL','KX','KZ','ZR','ZZ','ER6N','ER6F'],
          'DAYANG': ['DY1','DY2','DY5'],
          'SUNDOWN': ['WEB','FIFITY','PALIO','PGO','STX','VBLADE','EVO', 'HUNTER
MAX'],
          'SHINERAY':
['BIKE','BRAVO','DISCOVER','EAGLE','INDIANAPOLIS','JET','NEW','WAVE',
                       'STRONG','SUPER SMART','VENICE',' XY']}

# Função para pegar a chave pelo valor, dado que valor é único.
def pegaChave(v, dict):
    for chave, valores in dict.items():
        if type(valores)!=type([1,2]):
            valores=[valores]
        for valor in valores:
            if v == valor:
                return chave
    return "Não existe chave para esse valor."

def acrescentaMarca(descricao):
    for marca in Marcas:
        if re.search(marca,descricao.upper()):
            descricao += " "+marca
        for termo in Marcas[marca]:
            t1=termo.split()
            if len(t1)>1:
                pat=r"(?:"+t1[0]+r"|"+t1[1]+r").*(?:"+t1[0]+r"|"+t1[1]+r")"
            elif len(termo)<3:
                pat=termo+r"([0-9]{1,}|\b)"
            else:
                pat=termo
            resultados = re.findall(pat,descricao.upper())
            if resultados:
                descricao += " "+marca
                descricao += " "+" ".join(resultados)
                descricao += " "+termo
    descricao=" ".join(sorted(set(descricao.lower().split())))
    return descricao

# ### Função final que transforma a DESCRICAO DO PRODUTO em Modelo para
classificar
def criaModelo(descricao):
    descricao=limpaDescricao(descricao)
    descricao=achaPalavraChave(descricao)
    descricao=acrescentaMarca(descricao)
    return descricao

# ### Função que determina a existência de retentor no kit e retorna True|False
def retentorAux(descricao):
    # define o padrão de busca
    padrao = r'c/ *ret|com *ret' #r"(com|c/) *ret"
    descricao=descricao.lower()
    busca = re.findall(padrao, descricao)
    if busca:
        descricao = busca[0]
```

```
        return True
    else:
        descricao=''
        return False

# ## Carrega o modelo Linear SVC
# ### Carrega arquivos do pickle
with open(r'./pickle/clfsvc.pkl', 'rb') as file:
    clfsvc = pickle.load(file)
    file.close()
with open(r'./pickle/cvt.pkl', 'rb') as file:
    cvt = pickle.load(file)
    file.close()

# ### Define a função de classificação
def classificaAplicacao(descricao):
    modelo = criaModelo(descricao)
    novo_cvt = cvt.transform(pd.Series(modelo))
    aplicacao = clfsvc.predict(novo_cvt)[0]
    return aplicacao
```