

```

1 # funcoesTCC.py
2 #
3 #!/usr/bin/env python
4 # coding: utf-8
5
6 # # Classificação dos modelos de motocicleta a partir da descrição do produto
7
8 import pandas as pd, numpy as np
9 import re
10 import pickle
11
12 # ### Importa as stopwords da língua portuguesa
13 # Importar lista de Stopwords
14 from nltk.corpus import stopwords
15 stopwords = set(stopwords.words('portuguese'))
16 # Palavras a adicionar na lista de stopwords estão contidas em um arquivo csv
# externo
17 dfsw = pd.read_csv('./bases/stopwords.csv', encoding='ISO-8859-1')
18 stopwords_df=sorted(list(dfsw['stopword']))
19 # Atualizar stopwords
20 stopwords.update(stopwords_df)
21
22 # ### Carrega lista de aplicações e cria palavrasChave
23 # carrega a lista de marcas de motos do arquivo
24 dfaplicacoes=pd.read_csv('./bases/Aplicacoes.csv')
25 # remove caracteres especiais ou soltos e termos duplicados, salvando na lista
26 palavrasChave=sorted(set(re.sub(r"\b \w \b",
27                               ' ',
28                               re.sub(r"/>()|\+\$%#@\'\"]+",
29                               " ",
30                               ".join(dfaplicacoes['APLICACOES'].tolist()))).split())
31
32 # ##### Função de limpeza de dados irrelevantes para a classificação e remoção de
# stopwords
33 def limpaDescricao(descricao): #
34     descricao=descricao.lower() #transformar em minúsculas
35     # remove top (1045) e variantes
36     descricao=re.sub(r'\b[ (-]*top \(*1045 *\)[- ]*\b', ' ', descricao)
37     # remove códigos numéricos entre parênteses com -*/
38     descricao=re.sub(r"\(*\d*[/\/*\-\d]*\d*\*\)", ' ', descricao)
39     # remove a ocorrência de "código e etc." e o termo seguinte começado com
# número
40     # att: (alguns tem hífen ou asterisco) (colocar antes de remover pontuação)
41     descricao=re.sub(r"\b(invoice|código|codigo|cod|cód|(certificado|cert)(no|nr)|ref)[0-9a-z/\-\*\.\.:]* *\d[^ ]+", ' ', descricao)
42     # remove identificação de referência de engrenagens dos kits (antes da
# pontuação)
43     #descricao=re.sub(r"([^\w\s])|(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)
44     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
45     # substitui os termos "s/re" e "s/ret" por "sem retentor"
46     descricao=re.sub(r"\b(s\|re|s\|ret)\b", 'sem retentor', descricao)
47     # substitui os termos "c/re" ou "c/ret" por "com retentor"
48     descricao=re.sub(r"\b(c\|re|c\|ret)\b", 'com retentor', descricao)
49     # substitui o termo "aplicação" e "modelo" emendado com outro

```

```

50     descricao=re.sub(r"aplicacao", "aplicacao ", descricao)
51     descricao=re.sub(r"modelo", "modelo ", descricao)
52     # remove códigos no início da descrição
53     descricao=re.sub(r"\b\d{2,}[^ ]*\b", ' ', descricao)
54     descricao=re.sub(r"\^k[^ ]+", ' ', descricao)
55     descricao=re.sub(r"- | -|[\\]+,.;!/?]+", ' ', descricao) #remover pontuação
56     (att: "- " ou "-")
57     #correção de erros de digitação comuns
58     termos={'titan': ['titian','tita','tintan','tit'],
59               'honda': ['hond','hnda','hon'],
60               'twister': ['twist', 'twiste'],
61               'dafra kansas': ['dafra kan'],
62               'tenere': ['tener','tenerre'],
63               'broz': ['bros','bross'],
64               'titan 150': ['titan150'],
65               'broz 150':
66               ['bross125.', 'bros125.', 'broz125', 'bross150.', 'bros150.', 'broz150'],
67               'pop 100': ['pop100'],
68               'phoenix': ['phoeni', 'phenix'],
69               'c100': ['c 100']}
70     for termo in termos:
71         for termoerrado in termos[termo]:
72             descricao=re.sub(r"\b"+termoerrado+r"\b", termo, descricao)
73     descricao=re.sub(r"/>()|+\$\%#@'\\"]+", ' ', descricao) #remover
74     caracteres especiais
75     # remove a ocorrência de medidas tipo 00x000x00 ou 000x0000
76     descricao=re.sub(r"\b\d{1,}(x|*)\d{1,}(x|*)\d{1,}|\d{1,}(x|*)\d{1,}\b", ' ',
77     descricao)
78     # remove identificação de quantidades, unidades, peças e conjuntos
79     descricao=re.sub(r"\b(\d* *(conj|und|uni|pc|pc|pec|pec)( \w|\w)+?)\b", ' ',
80     descricao)
81     # remove identificação de mais de 4 dígitos com ou sem letras no início e no
82     final
83     descricao=re.sub(r"\w+\d{4,}\w+", ' ', descricao)
84     # remove números de 4 dígitos ou mais começados de 2 a 9
85     descricao=re.sub(r"\b[02-9]\d{3,}\b", ' ', descricao)
86     # remove identificação de termos começados por zero
87     descricao=re.sub(r"\b0\d*\w+?(?=\\b)", ' ', descricao)
88     # remove a ocorrência de "marca " e o termo na lista até o próximo espaço
89     for marca in ['kmc *gold', 'am *gold', 'king', 'bravo *racing', 'riffel *top']:
90         descricao=re.sub(r"\bmarca[ :./]*"+str(marca)+r"[^ ]*", ' ', descricao)
91     # colocar antes das stopwords
92     descricao=re.sub(r"marca[ :./]*\w+", ' ', descricao)
93     descricao=re.sub(r"(-| - )", ' ', descricao)
94     # remove stopwords mantendo a ordem original da descrição
95     descricao=list(dict.fromkeys(descricao.split())) # cria lista com termos
96     únicos
97     descricao=" ".join([x for x in descricao if x not in set(stopwords)]) #
98     exclui stopwords
99     # limpa os número que não estão na lista de aplicações (colocar depois das
100    stopwords)
101    desc=descricao.upper().split() # quebra a descrição
102    dif=list(set(descricao.upper().split()).difference(palavrasChave)) # pega os
103    termos diferentes de palavrasChave
104    [desc.remove(x) for x in desc if (x in dif and x.isnumeric())] # exclui de
105    desc os termos numéricos diferentes
106    descricao=" ".join(desc).lower() # volta para texto
107    #remover hífen, letras ou números soltos (deixar duplicado mesmo)
108    descricao=re.sub(r"(-| - )\b\w\b", ' ', descricao)

```

```

97     descricao=re.sub(r"^-| - |\b\w\b)", ' ', descricao)
98     #substitui remove o i das cilindradas: ex.: 125i por 125
99     termos=re.findall(r"\d{1,}i\b",descricao)
100    if termos:
101        for termo in termos:descricao=descricao.replace(termo,termo[:-1])
102        # remove espaços em excesso (colocar no final)
103        descricao=re.sub(r" {2,}", ' ', descricao)
104        descricao=descricao.strip()
105        # retorna a descricao como saída da função
106        return descricao # retorna a descrição
107
108 def achaPalavraChave(descricao):
109     palavras=[]
110     descricao=descricao.upper()
111     desc=descricao.split()
112     for palavra in palavrasChave:
113         if palavra in desc:
114             palavras.append(palavra)
115         else:
116             if palavra.isnumeric():
117                 pat=r"[0-9]*"+str(palavra)+r"[0-9]*"
118             elif palavra.isalpha():
119                 pat=r"[A-Z]*"+str(palavra)+r"[A-Z]*"
120             else:
121                 pat=r"\b"+palavra+r"\b"
122                 a = re.findall(pat,descricao)
123             if len(a)>0:
124                 # adiciona resultado nas palavras se o resultado estiver em
125                 palavras+=[a[i] for i in range(len(a)) if a[i] in palavrasChave]
126     palavras=list(set(palavras)) # remove duplicados
127     palavras=" ".join(palavras) # converte para string
128     return palavras.lower()
129
130 # termos que iniciam item da descrição correspondem a marca
131 # As que começam com espaço devem permanecer assim, pois há outros modelos com o
132 # mesmo final
133 Marcas = {'HONDA': ['CG', 'CD', 'CBX', 'CB', 'CBR', 'CRF', 'BIZ', 'BROS', 'BROZ', 'XL', 'FAN', 'XR', 'XRE',
134                                     'DREAM', 'TITAN', 'TODAY', 'TWIN', 'POP', 'NX', 'NXR', 'TWISTER',
135                                     'HORNET',
136                                     'AMERICA', 'BOLDOR', 'DUTY', 'FIREBLADE', 'FURY', 'WING', 'LEAD', 'MAGNA', 'NL',
137                                     'NC', 'NSR', 'NC', 'NXR', 'PACIFIC', 'COAST', 'SHADOW',
138                                     'STRADA', 'STUNNER', 'HAWK',
139                                     'SUPERBLACKBIRD', 'TORNADO', 'TURUNA', 'XRV', 'AFRICA', 'VALKYRIE', 'VARADERO',
140                                     'VFR', 'VLR', 'VTR', 'VTX', 'TRANSALP'],
141                                     'YAMAHA': ['AEROX', 'ALBA', 'AXIS', 'BWS', 'DRAG ', 'DT', 'FZ', 'FJ', 'RD',
142                                     'TENERE',
143                                     'MT', 'XF', 'XJ', 'XS', 'XT', 'XZ', 'YF', 'YZ', 'LANDER', 'GLADIATOR', 'GRIZZLY',
144                                     'YBR', 'YZ', 'VIRAGO', 'FACTOR', 'EC', 'CRYPTON', 'FAZER', 'JOG',
145                                     'LANDER',
146                                     'FROG', 'LIBERO', 'MAJESTY', 'MEST', 'MIDNIGHT', 'MORPH', 'NEO', 'PASSOL'],
147                                     'DAFRA':
148                                     ['APACHE', 'CITYCOM', 'KANSAS', 'LASER', 'NEXT', 'RIVA', 'ROADWIN', 'ZIG', 'SPEED'],
149                                     'SUZUKI': ['KATANA', 'YES', 'INTRUDER'],
150

```

```

144     'ZONGSHEN': ['ZS'],
145     'KASINSKI': ['COMET', 'MIRAGE'],
146     'POLARIS': ['SPORTSMAN', 'RZR', 'RANGER'],
147     'KAWASAKI':
148     ['NINJA', 'VERSYS', 'VOYAGER', 'GTR', 'KDX', 'KL', 'KX', 'KZ', 'ZR', 'ZZ', 'ER6N', 'ER6F'],
149     'DAYANG': ['DY1', 'DY2', 'DY5'],
150     'SUNDOWN': ['WEB', 'FIFTY', 'PALIO', 'PGO', 'STX', 'VBLADE', 'EVO', 'HUNTER
151 MAX'],
152     'SHINERAY':
153     ['BIKE', 'BRAVO', 'DISCOVER', 'EAGLE', 'INDIANAPOLIS', 'JET', 'NEW', 'WAVE',
154     'STRONG', 'SUPER SMART', 'VENICE', 'XY']}
155
156 # Função para pegar a chave pelo valor, dado que valor é único.
157 def pegaChave(v, dict):
158     for chave, valores in dict.items():
159         if type(valores)!=type([1,2]):
160             valores=[valores]
161         for valor in valores:
162             if v == valor:
163                 return chave
164
165     return "Não existe chave para esse valor."
166
167 def acrescentaMarca(descricao):
168     for marca in Marcas:
169         if re.search(marca,descricao.upper()):
170             descricao += " "+marca
171         for termo in Marcas[marca]:
172             t1=termo.split()
173             if len(t1)>1:
174                 pat=r"(?:" +t1[0]+r")|"+t1[1]+r").*(?:" +t1[0]+r"|" +t1[1]+r")"
175             elif len(termo)<3:
176                 pat=termo+r"([0-9]{1,})|\b\b"
177             else:
178                 pat=termo
179             resultados = re.findall(pat,descricao.upper())
180             if resultados:
181                 descricao += " "+marca
182                 descricao += " "+"".join(resultados)
183                 descricao += " "+termo
184             descricao=" ".join(sorted(set(descricao.lower().split())))
185
186     return descricao
187
188 # ### Função final que transforma a DESCRIÇÃO DO PRODUTO em Modelo para
189 # classificar
190 def criaModelo(descricao):
191     descricao=limpaDescricao(descricao)
192     descricao=achaPalavraChave(descricao)
193     descricao=acrescentaMarca(descricao)
194
195     return descricao
196
197 # ### Função que determina a existência de retentor no kit e retorna True|False
198 def retentorAux(descricao):
199     # define o padrão de busca
200     padrao = r'c/ *ret|com *ret' #r"(com|c/) *ret"
201     descricao=descricao.lower()
202     busca = re.findall(padrao, descricao)
203     if busca:
204         descricao = busca[0]

```

```
198     return True
199 else:
200     descricao=''
201     return False
202
203 # ## Carrega o modelo Linear SVC
204 # ### Carrega arquivos do pickle
205 with open(r'./pickle/clfsvc.pkl', 'rb') as file:
206     clfsvc = pickle.load(file)
207     file.close()
208 with open(r'./pickle/cvt.pkl', 'rb') as file:
209     cvt = pickle.load(file)
210     file.close()
211
212 # ### Define a função de classificação
213 def classificaAplicacao(descricao):
214     modelo = criaModelo(descricao)
215     novo_cvt = cvt.transform(pd.Series(modelo))
216     aplicacao = clfsvc.predict(novo_cvt)[0]
217     return aplicacao
```

```

1 # prettyPlotConfusionMatrix.py
2 #
3 # -*- coding: utf-8 -*-
4 """
5 plot a pretty confusion matrix with seaborn
6 Created on Mon Jun 25 14:17:37 2018
7 @author: Wagner Cipriano - wagnerbhbr - gmail - CEFETMG / MMC
8 Download: https://github.com/wcipriano/pretty-print-confusion-matrix
9 REFERENCES:
10   https://www.mathworks.com/help/nnet/ref/plotconfusion.html
11   https://stackoverflow.com/questions/28200786/how-to-plot-scikit-learn-
12     classification-report
13   https://stackoverflow.com/questions/5821125/how-to-plot-confusion-matrix-
14     with-string-axis-rather-than-integer-in-python
15   https://www.programcreek.com/python/example/96197/seaborn.heatmap
16   https://stackoverflow.com/questions/19233771/sklearn-plot-confusion-matrix-
17     with-labels/31720054
18   http://scikit-
19     learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-
20     glr-auto-examples-model-selection-plot-confusion-matrix-py
21 """
22
23
24 #imports
25 from pandas import DataFrame
26 import numpy as np
27 import matplotlib.pyplot as plt
28 import matplotlib.font_manager as fm
29 from matplotlib.collections import QuadMesh
30 import seaborn as sn
31
32
33 def get_new_fig(fn, figsize=[9,9]):
34     """
35     Init graphics
36     """
37     fig1 = plt.figure(fn, figsize)
38     ax1 = fig1.gca()    #Get Current Axis
39     ax1.cla() # clear existing plot
40     return fig1, ax1
41
42
43 def configcell_text_and_colors(array_df, lin, col, oText, facecolors, posi, fz,
44 fmt, show_null_values=0):
45     """
46         config cell text and colors
47         and return text elements to add and to dell
48         @TODO: use fmt
49     """
50
51     text_add = []; text_del = [];
52     cell_val = array_df[lin][col]
53     tot_all = array_df[-1][-1]
54     per = (float(cell_val) / tot_all) * 100
55     curr_column = array_df[:,col]
56     ccl = len(curr_column)
57
58     #last line and/or last column
59     if(col == (ccl - 1)) or (lin == (ccl - 1)):
60         #tots and percents
61         if(cell_val != 0):

```

```

52     if(col == ccl - 1) and (lin == ccl - 1):
53         tot_rig = 0
54         for i in range(array_df.shape[0] - 1):
55             tot_rig += array_df[i][i]
56             per_ok = (float(tot_rig) / cell_val) * 100
57     elif(col == ccl - 1):
58         tot_rig = array_df[lin][lin]
59         per_ok = (float(tot_rig) / cell_val) * 100
60     elif(lin == ccl - 1):
61         tot_rig = array_df[col][col]
62         per_ok = (float(tot_rig) / cell_val) * 100
63         per_err = 100 - per_ok
64     else:
65         per_ok = per_err = 0
66
67     if per_ok ==100:
68         per_ok_s ='%.2f%%' %(per_ok)
69     else:
70         per_ok_s = '100%'
71
72     #text to DEL
73     text_del.append(oText)
74
75     #text to ADD
76     font_prop = fm.FontProperties(weight='bold', size=fz)
77     text_kwarg = dict(color='w', ha="center", va="center", gid='sum',
fontproperties=font_prop)
78     lis_txt = ['%d'%(cell_val), per_ok_s, '%.2f%%' %(per_err)]
79     lis_kwa = [text_kwarg]
80     dic = text_kwarg.copy(); dic['color'] = 'g'; lis_kwa.append(dic);
81     dic = text_kwarg.copy(); dic['color'] = 'r'; lis_kwa.append(dic);
82     lis_pos = [(oText._x, oText._y-0.3), (oText._x, oText._y), (oText._x,
oText._y+0.3)]
83     for i in range(len(lis_txt)):
84         newText = dict(x=lis_pos[i][0], y=lis_pos[i][1], text=lis_txt[i],
kw=lis_kwa[i])
85         #print 'lin: %s, col: %s, newText: %s' %(lin, col, newText)
86         text_add.append(newText)
87         #print '\n'
88
89     #set background color for sum cells (last line and last column)
90     carr = [0.27, 0.30, 0.27, 1.0]
91     if(col == ccl - 1) and (lin == ccl - 1):
92         carr = [0.17, 0.20, 0.17, 1.0]
93     facecolors[posi] = carr
94
95     else:
96         if(per > 0):
97             txt = '%s\n%.2f%%' %(cell_val, per)
98         else:
99             if(show_null_values == 0):
100                 txt = ''
101             elif(show_null_values == 1):
102                 txt = '0'
103             else:
104                 txt = '0\n0.0%'
105             oText.set_text(txt)

```

```

106
107     #main diagonal
108     if(col == lin):
109         #set color of the textin the diagonal to white
110         oText.set_color('w')
111         # set background color in the diagonal to blue
112         facecolors[posi] = [0.35, 0.8, 0.55, 1.0]
113     else:
114         oText.set_color('r')
115
116     return text_add, text_del
117 #
118
119 def insert_totals(df_cm):
120     """ insert total column and line (the last ones) """
121     sum_col = []
122     for c in df_cm.columns:
123         sum_col.append( df_cm[c].sum() )
124     sum_lin = []
125     for item_line in df_cm.iterrows():
126         sum_lin.append( item_line[1].sum() )
127     df_cm['sum_lin'] = sum_lin
128     sum_col.append(np.sum(sum_lin))
129     df_cm.loc['sum_col'] = sum_col
130     #print ('\n df_cm:\n', df_cm, '\n\n')
131 #
132
133 def pretty_plot_confusion_matrix(df_cm, annot=True, cmap="Oranges", fmt='.2f',
134 , fz=11,
135 , lw=0.5, cbar=False, figsize=[8,8], show_null_values=0,
136 pred_val_axis='y',insertTot=True):
137     """
138         print conf matrix with default layout (like matlab)
139         params:
140             df_cm           dataframe (pandas) without totals
141             annot          print text in each cell
142             cmap           Oranges,Oranges_r,YlGnBu,Blues,RdBu, ... see:
143             fz            fontsize
144             lw            linewidth
145             pred_val_axis where to show the prediction values (x or y axis)
146                             'col' or 'x': show predicted values in columns (x axis)
147             instead lines           'lin' or 'y': show predicted values in lines   (y axis)
148             """
149             if(pred_val_axis in ('col', 'x')):
150                 xlabel = 'Predição'
151                 ylabel = 'Classificação'
152             else:
153                 xlabel = 'Classificação'
154                 ylabel = 'Predição'
155                 df_cm = df_cm.T
156
157                 # create "Total" column
158                 insert_totals(df_cm)
159
160                 #this is for print allways in the same window
161                 fig, ax1 = get_new_fig('Conf matrix default', figsize)

```

```

160
161     #thanks for seaborn
162     if insertTot:
163         ax = sn.heatmap(df_cm, annot=annot, annot_kws={"size": fz},
164                           linewidths=lw, ax=ax1,
165                           cbar=cbar, cmap=cmap, linecolor='w', fmt=fmt)
166     else:
167         ax = sn.heatmap(df_cm.iloc[:-1,:-1], annot=annot, annot_kws={"size": fz},
168                           linewidths=lw, ax=ax1,
169                           cbar=cbar, cmap=cmap, linecolor='w', fmt=fmt)
170
171     #set ticklabels rotation
172     ax.set_xticklabels(ax.get_xticklabels(), rotation = 90, fontsize = 10)
173     ax.set_yticklabels(ax.get_yticklabels(), rotation = 0, fontsize = 10)
174
175     # Turn off all the ticks
176     for t in ax.xaxis.get_major_ticks():
177         t.tick1On = False
178         t.tick2On = False
179     for t in ax.yaxis.get_major_ticks():
180         t.tick1On = False
181         t.tick2On = False
182
183     #face colors list
184     quadmesh = ax.findobj(QuadMesh)[0]
185     facecolors = quadmesh.get_facecolors()
186
187     #iter in text elements
188     array_df = np.array( df_cm.to_records(index=False).tolist() )
189     text_add = []; text_del = [];
190     posi = -1 #from left to right, bottom to top.
191     for t in ax.collections[0].axes.texts: #ax.texts:
192         pos = np.array( t.get_position() ) - [0.5,0.5]
193         lin = int(pos[1]); col = int(pos[0]);
194         posi += 1
195         #print ('>>> pos: %s, posi: %s, val: %s, txt: %s' %(pos, posi,
196         array_df[lin][col], t.get_text()))
197
198         #set text
199         txt_res = configcell_text_and_colors(array_df, lin, col, t, facecolors,
200         posi, fz, fmt, show_null_values)
201
202         text_add.extend(txt_res[0])
203         text_del.extend(txt_res[1])
204
205     #remove the old ones
206     for item in text_del:
207         item.remove()
208     #append the new ones
209     for item in text_add:
210         ax.text(item['x'], item['y'], item['text'], **item['kw'])
211
212     #titles and legends
213     ax.set_title('Matriz de Confusão', fontsize = 30)
214     ax.set_xlabel(xlbl, fontsize = 25)
215     ax.set_ylabel(ylbl, fontsize = 25)
216     plt.tight_layout() #set layout slim
217     plt.show()

```


Notebook Jupyter 1a_trataCSVsSiscori

Importação e tratamento dos dados Siscori

Os dados importados no Siscore vêm em arquivos do tipo CSV no formato **CAPINNAAMM**, onde NN é o número do capítulo extraído, AA é o ano com dois dígitos e MM é o mês com dois dígitos, formando o período de referência dos dados extraídos. O arquivo CSV obtido vem configurado com o separador "@" e descrição da coluna com excesso de espaços, o que precisa de uma camada de tratamento para correta importação dos dados.

Importação das Bibliotecas

In [1]:

```
import pandas as pd
import numpy as np
import os, time
```

In [2]:

```
# Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
```

Código executado em 16/01/2022 às 17:49

Definindo dados gerais

Cria lista de arquivos CSV contidos na pasta atual

In [3]:

```
arqsCSV = []
for arquivo in os.listdir("./bases/siscori/"):
    if arquivo[-3:].upper() == "CSV" and arquivo[:4] == 'CAPI':
        arqsCSV.append("./bases/siscori/" + arquivo)
arqsCSV=sorted(arqsCSV)
print(arqsCSV)

 ['./bases/siscori/CAPI872001.CSV', './bases/siscori/CAPI872002.CSV', './bases/siscori/CAPI872003.CSV', './bases/siscori/CAPI872004.CSV', './bases/siscori/CAPI872005.CSV', './bases/siscori/CAPI872006.CSV', './bases/siscori/CAPI872007.CSV', './bases/siscori/CAPI872008.CSV', './bases/siscori/CAPI872009.CSV', './bases/siscori/CAPI872010.CSV', './bases/siscori/CAPI872011.CSV', './bases/siscori/CAPI872012.CSV', './bases/siscori/CAPI872101.CSV', './bases/siscori/CAPI872102.CSV', './bases/siscori/CAPI872103.CSV', './bases/siscori/CAPI872104.CSV', './bases/siscori/CAPI872105.CSV', './bases/siscori/CAPI872106.CSV']
```

Cria variáveis com nomes das colunas e seus tipos

In [4]:

```
tipos = {'NUMERO DE ORDEM': str,
         'ANOMES': str,
         'COD.NCM': str,
         'DESCRICAO DO CODIGO NCM': object,
         'PAIS.OR': int,
         'PAIS DE ORIGEM': object,
         'PAIS.AQ': int,
         'PAIS DE AQUISICAO': object,
         'UND.ESTAT.': int,
         'UNIDADE DE MEDIDA': object,
         'UNIDADE COMERC.': object,
         'DESCRICAO DO PRODUTO': object,
         'QTDE ESTATISTICA': float,
         'PESO LIQUIDO': float,
         'VLE DOLAR': float,
         'VL FRETE DOLAR': float,
         'VL SEGURO DOLAR': float,
         'VALOR UN.PROD.DOLAR': float,
         'QTD COMERCIAL.': float,
         'TOT.UN.PROD.DOLAR': float,
         'UNIDADE DESEMBARQUE': object,
         'UNIDADE DESEMBARACO': object,
         'INCOTERM': object,
         'NAT.INFORMACAO': object,
         'SITUACAO DO DESPACHO': object}
colunas = list(tipos.keys())
```

Inicializa a variável que conterá o tamanho total do dataset original

In [5]:

```
tamanhoDataset=0
```

Inicializa um dataframe vazio que conterá os dados finais

In [6]:

```
df = pd.DataFrame(columns = colunas)  
df.head()
```

Out[6]:

NUMERO DE ORDEM	ANOMES	COD.NCM	DESCRICAO DO CODIGO NCM	PAIS.OR	PAIS DE ORIGEM	PAIS.AQ
-----------------------	--------	---------	-------------------------------	---------	-------------------	---------

0 rows × 25 columns

Importa cada CSV, trata e concatena no DataFrame final

In [7]:

```
# Executa para cada CSV na lista
for arqCSV in arqsCSV:
    print('Iniciando ' + arqCSV + '.')
    dftemp = pd.read_csv(arqCSV,
                         sep='@',
                         decimal=r',',
                         engine='python',
                         encoding = "ISO-8859-1",
                         header = 0,
                         names = colunas,
                         dtype = tipos,
                         quotechar="",
                         error_bad_lines=False,
                         warn_bad_lines=False)
    print('DataFrame carregado...\nAplicando filtros...')
    # Elimina os registros sem valores ou nulos
    dftemp = dftemp.dropna()
    # Incrementa o tamanho do Dataset
    tamanhoDataset += dftemp[dftemp.columns[0]].count()
    print('Dados da importação do arquivo')
    print('Quantidade de registros válidos:' + f'{str(dftemp[dftemp.columns[0]].count()):>8}')
    # Filtra o DataFrame somente com os registros de interesse
    # Filtro 1: NCM de interesse: 87141000
    indiceNCM = dftemp['COD.NCM'] == '87141000'
    dftemp = dftemp[indiceNCM]
    # Filtro 2: Incluir registros com descrição contendo palavras da Lista
    # a incluir
    listafiltroincluir = ["transm", "corrente", "coroa", "pinhao|pinhão"] # A barra vertical (/) faz o "ou".
    for termo in listafiltroincluir:
        dftemp=dftemp[dftemp['DESCRICA DO PRODUTO'].str.contains(termo, case=False)]
    # Filtro 3: Excluir registros com descrição contendo palavras da Lista
    padraofiltroexcluir = r"semi|reposicao|reposição"
    dftemp=dftemp[dftemp['DESCRICA DO PRODUTO'].str.contains(padraofiltroe
xcluir, case=False, regex=True)==False]
    print(f'Quantidade de registros filtrados:' + f'{str(dftemp[dftemp.colu
mnos[0]].count()):>6}')
    # Concatena com o DataFrame final
    df = pd.concat([df,dftemp])
    print(arqCSV + ' finalizado.\n')
```

```
Iniciando ./bases/siscori/CAPI872001.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 491150  
Quantidade de registros filtrados: 810  
../bases/siscori/CAPI872001.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872002.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 376497  
Quantidade de registros filtrados: 681  
../bases/siscori/CAPI872002.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872003.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 435878  
Quantidade de registros filtrados: 906  
../bases/siscori/CAPI872003.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872004.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 330218  
Quantidade de registros filtrados: 184  
../bases/siscori/CAPI872004.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872005.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 247533  
Quantidade de registros filtrados: 429  
../bases/siscori/CAPI872005.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872006.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 230660  
Quantidade de registros filtrados: 625  
../bases/siscori/CAPI872006.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872007.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 305473  
Quantidade de registros filtrados: 1236  
../bases/siscori/CAPI872007.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872008.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 340985  
Quantidade de registros filtrados: 1395  
../bases/siscori/CAPI872008.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872009.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 334556  
Quantidade de registros filtrados: 1421  
../bases/siscori/CAPI872009.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872010.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 431598  
Quantidade de registros filtrados: 1167  
../bases/siscori/CAPI872010.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872011.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 435236  
Quantidade de registros filtrados: 1084  
../bases/siscori/CAPI872011.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872012.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 464189  
Quantidade de registros filtrados: 1009  
../bases/siscori/CAPI872012.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872101.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 463810  
Quantidade de registros filtrados: 1278  
../bases/siscori/CAPI872101.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872102.CSV.  
DataFrame carregado...  
Aplicando filtros...  
Dados da importação do arquivo  
Quantidade de registros válidos: 407665  
Quantidade de registros filtrados: 1594  
../bases/siscori/CAPI872102.CSV finalizado.
```

```
Iniciando ./bases/siscori/CAPI872103.CSV.
```

```
DataFrame carregado...
Aplicando filtros...
Dados da importação do arquivo
Quantidade de registros válidos: 549879
Quantidade de registros filtrados: 1517
./bases/siscori/CAPI872103.CSV finalizado.

Iniciando ./bases/siscori/CAPI872104.CSV.
DataFrame carregado...
Aplicando filtros...
Dados da importação do arquivo
Quantidade de registros válidos: 636066
Quantidade de registros filtrados: 1155
./bases/siscori/CAPI872104.CSV finalizado.

Iniciando ./bases/siscori/CAPI872105.CSV.
DataFrame carregado...
Aplicando filtros...
Dados da importação do arquivo
Quantidade de registros válidos: 482319
Quantidade de registros filtrados: 877
./bases/siscori/CAPI872105.CSV finalizado.

Iniciando ./bases/siscori/CAPI872106.CSV.
DataFrame carregado...
Aplicando filtros...
Dados da importação do arquivo
Quantidade de registros válidos: 729922
Quantidade de registros filtrados: 908
./bases/siscori/CAPI872106.CSV finalizado.
```

Resetando o índice do DataFrame importado

In [8]:

```
df.reset_index(inplace=True, drop=True)
```

Excluindo colunas desnecessárias

In [9]:

```
excluir=[ 'NUMERO DE ORDEM',
          'ANOMES',
          'COD.NCM',
          'DESCRICAO DO CODIGO NCM',
          'PAIS.OR',
          'PAIS.AQ',
          'PAIS DE AQUISICAO',
          'UND.ESTAT.',
          'UNIDADE DE MEDIDA',
          'UNIDADE COMERC.',
          'QTDE ESTATISTICA',
          'PESO LIQUIDO',
          'VOLUME DOLAR',
          'VL FRETE DOLAR',
          'VL SEGURO DOLAR',
          'QTD COMERCIAL.',
          'TOT.UN.PROD.DOLAR',
          'UNIDADE DESEMBARQUE',
          'UNIDADE DESEMBARACO',
          'INCOTERM',
          'NAT.INFORMACAO',
          'SITUACAO DO DESPACHO']
```

In [10]:

```
df=df.drop(excluir, axis=1)
```

Remover espaços em excesso nos campos string

In [11]:

```
# Remove espaços em excesso das colunas em colstr
colstr = ['PAIS DE ORIGEM',
          'DESCRICAO DO PRODUTO']
for coluna in colstr:
    df[coluna]=df[coluna].str.strip()
```

Verificando o DataFrame importado

In [12]:

```
df.sample(5)
```

Out[12]:

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR
6916	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO P MOTOCICLETA MOD.: FAZER 1...	5.730000
3841	CHINA, REPUBLICA POP	22292 - 91083 - KIT DE TRANSMISSAO PARA MOTOCI...	8.680000
6339	CHINA, REPUBLICA POP	878192 - KIT DE TRANSMISSÃO, COMPOSTO DE CORRE...	22.214749
7211	CHINA, REPUBLICA POP	007266# KIT TRANSMISSÃO STANDARD TEMPERADO COM...	5.201000
5612	CHINA, REPUBLICA POP	TRANSMISSAO PARA USO EM MOTOCICLETA COMPOSTO D...	3.560000

In [13]:

```
df.shape
```

Out[13]:

(18276, 3)

Exportando o DataFrame

Exportando para um arquivo CSV

In [14]:

```
df.to_csv(r'./bases/dataframe.csv', index = False, header = True)
```

Exportando para um arquivo de planilha do Excel

In [15]:

```
df.to_excel(r'./bases/dataframe.xlsx', index = False, header = True)
```

Compara o tamanho total do Dataset inicial e final

In [16]:

```
print('\nQuantidade total de registros válidos importados: ' + f'{str(tamanhoDataset):>8}')
print('Tamanho do dataset após aplicação dos filtros:    ' + f'{str(df.shape[0]):>8}')
```

Quantidade total de registros válidos importados: 7693634
Tamanho do dataset após aplicação dos filtros: 18276

In [17]:

```
tempotot=time.time()-initot
if tempotot>60:
    print(f'Tempo total de execução: {tempotot/60:.2f} minutos.')
else:
    print(f'Tempo total de execução: {tempotot:.2f} segundos.')
```

Tempo total de execução: 5.42 minutos.

Notebook Jupyter 1b_trataABIMOTO13

Importação e tratamento Tabela ABIMOTO v13

A tabela ABIMOTO v13 é fornecida em PDF, que foi manualmente transformado em uma planilha do Excel no formato XLSX.

Importação das Bibliotecas

In [1]:

```
import pandas as pd
import numpy as np
import os, time
```

In [2]:

```
# Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
```

Código executado em 16/01/2022 às 18:00

Definindo dados gerais

Cria variáveis com nomes das colunas e seus tipos

In [3]:

```
tipos = { 'ITEM' : int,
          'PARTES E PEÇAS' : str,
          'MOTO PARTS' : str,
          'NCM' : str,
          'UNI' : str,
          'VMLE' : float }
colunas = list(tipos.keys())
```

Inicializa a variável que conterá o tamanho total do dataset original

In [4]:

```
tamanhoDataset=0
```

Inicializa um dataframe vazio que conterá os dados finais

In [5]:

```
df = pd.DataFrame(columns = colunas)
df.head()
```

Out[5]:

ITEM	PARTES E PEÇAS	MOTO PARTS	NCM	UNI	VMLE
------	----------------	------------	-----	-----	------

Importa o arquivo Excel XLSX, trata e concatena no DataFrame final

In [6]:

```
# Executa a importação e tratamento do arquivo
arq = './bases/ABIMOT013.xlsx'
print('Iniciando ' + arq + '.')
df = pd.read_excel(arq,
                    sep='@',
                    decimal=r',',
                    encoding = "ISO-8859-1",
                    header = 0,
                    names = colunas,
                    dtype = tipos,
                    quotechar="''",
                    error_bad_lines=False)
print('DataFrame carregado...\nAplicando filtros...')
# Elimina os registros sem valores ou nulos
df = df.dropna()
# Incrementa o tamanho do Dataset
tamanhoDataset += df[df.columns[0]].count()
print('Quantidade de registros válidos: ' + str(tamanhoDataset))
# Filtra o DataFrame somente com os registros de interesse
# Filtro 1: NCM de interesse: 87141000
indiceNCM = df['NCM'] == '87141000'
df = df[indiceNCM]
# Filtro 2: Descrição contendo 'kit' e 'transm'
df = df[df['PARTES E PEÇAS'].str.contains("kit", case=False)]
df = df[df['PARTES E PEÇAS'].str.contains("transm", case=False)]
print(arq + ' finalizado.\n')
```

Iniciando ./bases/ABIMOT013.xlsx.
 DataFrame carregado...
 Aplicando filtros...
 Quantidade de registros válidos: 129
 ./bases/ABIMOT013.xlsx finalizado.

Ajustando a descrição e indicação de presença de corrente com retentor no kit

In [7]:

```
df.head()
```

Out[7]:

	ITEM	PARTES E PEÇAS	MOTO PARTS	NCM	UNI	V
77	78	KIT TRANSMISSÃO 1045 C100 BIZ 15D+428HX108+35D...	TRANS.SET 1045 C100 BIZ 15D+428HX108+35D W/PR.	87141000	RETENTOR KIT COM	
78	79	KIT TRANSMISSÃO 1045 C100 BIZ 15D+428HX108+35D...	TRANS.SET 1045 C100 BIZ 15D+428HX108+35D W/PR.	87141000	RETENTOR KIT SEM	
79	80	KIT TRANSMISSÃO 1045 (CORRENTE.COROA. PINHAO) ...	TRANS.SET 1045 (CHAIN + SPROCKET) CBX 250 TWISTER	87141000	KIT COM RETENTOR	
80	81	KIT TRANSMISSÃO 1045 (CORRENTE.COROA. PINHAO) ...	TRANS.SET 1045 (CHAIN + SPROCKET) CBX 250 TWISTER	87141000	KIT SEM RETENTOR	
81	82	KIT TRANSMISSÃO 1045 FAZER 250	TRANSMISSION SET + H CHAIN FAZER 251	87141000	RETENTOR KIT COM	

Em virtude de haver observação se há ou não corrente com retentor na coluna UND faremos a adequação da descrição pra conter tal informação.

In [8]:

```
obs = ["COM RETENTOR" if "COM" in x else "SEM RETENTOR" for x in df.UNI.str
      .split()]
df["PARTES E PEÇAS"] = df["PARTES E PEÇAS"] + " " + obs
```

Em seguida define-se a coluna UNI como 'KIT'

In [9]:

```
df['UNI'] = 'KIT'
```

E por fim cria-se a coluna RETENTOR do tipo *boolean* para indicar ou não a presença do retentor no kit.

In [10]:

```
from funcoesTCC import retentorAux # função importada do módulo de funções
criarModelo
df['RETENTOR'] = df['PARTES E PEÇAS'].apply(retentorAux)
```

Resetando o índice do DataFrame importado

In [11]:

```
df.reset_index(inplace=True, drop=True)
```

Excluindo colunas desnecessárias

In [12]:

```
excluir=['ITEM',
         'MOTO PARTS',
         'NCM',
         'UNI']
```

In [13]:

```
df=df.drop(excluir, axis=1)
```

Verificando o DataFrame final

In [14]:

```
df.head(3)
```

Out[14]:

	PARTES E PEÇAS	VMLE	RETENTOR
0	KIT TRANSMISSÃO 1045 C100 BIZ 15D+428HX108+35D...	6.47	True
1	KIT TRANSMISSÃO 1045 C100 BIZ 15D+428HX108+35D...	3.90	False
2	KIT TRANSMISSÃO 1045 (CORRENTE.COROA. PINHAO) ...	9.30	True

In [15]:

```
df.tail(3)
```

Out[15]:

		PARTES E PEÇAS	VMLE	RETENTOR
13	KIT TRANSMISSÃO XRE - 300 SEM RETENTOR	6.0	False	
14	KIT TRANSMISSÃO XTZ - 250 LANDER COM RETENTOR	8.2	True	
15	KIT TRANSMISSÃO XTZ - 250 LANDER SEM RETENTOR	4.4	False	

In [16]:

```
df.shape
```

Out[16]:

```
(16, 3)
```

Exportando o DataFrame

Exportando para um arquivo CSV

In [17]:

```
df.to_csv(r'./bases/dfABIMOTOv13.csv', index = False, header = True)
```

Exportando para um arquivo de planilha do Excel

In [18]:

```
df.to_excel(r'./bases/dfABIMOTOv13.xlsx', index = False, header = True)
```

Compara o tamanho total do Dataset inicial e final

In [19]:

```
print('Qtd de registros Dataset original: ' + str(tamanhoDataset))
print('Qtd de registros Dataset final:    ' + str(df[df.columns[0]].count()))
```

Qtd de registros Dataset original: 129
 Qtd de registros Dataset final: 16

In [20]:

```
tempotot=time.time()-initot
if tempotot>60:
    print(f'Tempo total de execução: {tempotot/60:.2f} minutos.')
else:
    print(f'Tempo total de execução: {tempotot:.2f} segundos.')
```

Tempo total de execução: 1.73 segundos.

Notebook Jupyter 2_geraWordclouds

Geração das WordClouds das descrições

Os dados gerados após o tratamento, objeto de estudo, estão contidos em um campo de descrição, onde o importador ou seu representante faz a entrada livre de dados, sem qualquer exigência ou padronização, nossa análise exploratória foi convincente no sentido de que deveria ser feito um tratamento com a utilização de Programação de Linguagem Natural – PLN. Dentro dessa análise exploratória de dados, observou-se a ocorrência de palavras que, embora frequentes, serão irrelevantes para a determinação da diferenciação entre os itens que estão classificados dentro do agrupamento representado pela NCM em estudo. Comum no estudo de categorização de textos, o uso de stopwords é recomendado nesse caso, pois da mesma forma que artigos, pronomes e outras palavras tradicionalmente identificadas como não relevantes para a distinção, termos como kit, transmissão, corrente, coroa e pinhão também são igualmente irrelevantes para distinguir um item de outro. Na biblioteca Natural Language Processing Toolkit - nltk já existe uma lista de stopwords para a língua portuguesa. Como essa biblioteca já possui a funcionalidade de complementação dessa lista, adicionou-se os itens kit, transmissão, corrente, coroa e pinhão; termos que pela sua característica não distinguem os itens dentro do dataset estudado.

Importa os dados já tratados

In [1]:

```
import pandas as pd
import time
```

In [2]:

```
# Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
```

Código executado em 20/01/2022 às 17:14

In [3]:

```
# Importa base de dados para um dataframe
df = pd.read_excel(r'./bases/dataframe.xlsx')
```

In [4]:

```
# Verifica o tamnanho do dataframe
df.shape
```

Out[4]:

(18276, 3)

In [5]:

```
# Mostra Linhas de exemplo do dataframe
df.sample(5)
```

Out[5]:

	PAIS DE ORIGEM	DESCRICAQ DO PRODUTO	VALOR UN.PROD.DOLAR
16225	CHINA, REPUBLICA POP	10530041 IN KIT TRANSMISSAO P/MOTOCICLETAS(COR... P/MOTOCICLETAS(COR...	4.3750
10597	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO (1045) COMPOSTO DE CORRENTE...	3.8710
10164	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO , MARCA RIFFEL, TITANIUM (1...	4.5657
12269	CHINA, REPUBLICA POP	item 05;Partes e peças para Motocicletas,Desta...	4.7022
15584	CHINA, REPUBLICA POP	ENGRENAGENS PARA TRANSMISSÃO DE MOTOCICLETAS E...	6.3900

Importa as stopwords da língua portuguesa

In [6]:

```
# Importar Lista de Stopwords
from nltk.corpus import stopwords
stopwords = set(stopwords.words('portuguese'))
```

In [7]:

```
# Mostra tamanho da Lista de stopwords
len(stopwords)
```

Out[7]:

204

In [8]:

```
# Mostra toda a lista de stopwords
swtemp = list(stopwords)
swtemp.sort()
print(swtemp)
```

```
['a', 'ao', 'aos', 'aqueла', 'aqueлas', 'aqueле', 'aqueлes',
'aquilo', 'as', 'at ', 'com', 'como', 'da', 'das', 'de', 'del
a', 'delas', 'dele', 'deles', 'depois', 'do', 'dos', 'e', 'el
a', 'elas', 'ele', 'eles', 'em', 'entre', 'era', 'eram', 'ess
a', 'essas', 'esse', 'esses', 'esta', 'estamos', 'estas', 'est
ava', 'estavam', 'este', 'esteja', 'estejam', 'estejamos', 'es
tes', 'esteve', 'estive', 'estivemos', 'estiver', 'estivera',
'estiveram', 'estiverem', 'estivermos', 'estivesse', 'estivess
em', 'estiv ramos', 'estiv ssimos', 'estou', 'est ', 'est vamo
s', 'est o', 'eu', 'foi', 'fomos', 'for', 'fora', 'foram', 'fo
rem', 'formos', 'fosse', 'fossem', 'fui', 'f ramos', 'f ssimo
s', 'haja', 'hajam', 'hajamos', 'havemos', 'hei', 'houve', 'ho
uvemos', 'houver', 'houvera', 'houveram', 'houverei', 'houvere
m', 'houveremos', 'houveria', 'houveriam', 'houvermos', 'houve
r ', 'houver o', 'houver amos', 'houvesse', 'houvessem', 'houv
 ramos', 'houv ssimos', 'h ', 'h o', 'isso', 'isto', 'j ', 'l 
e', 'lhes', 'mais', 'mas', 'me', 'mesmo', 'meu', 'meus', 'minh
a', 'minhas', 'muito', 'na', 'nas', 'nem', 'no', 'nos', 'noss
a', 'nossas', 'noso', 'nossos', 'num', 'numa', 'n o', 'n s',
'o', 'os', 'ou', 'para', 'pela', 'pelas', 'pelo', 'pelos', 'po
r', 'qual', 'quando', 'que', 'quem', 'se', 'seja', 'sejam', 's
ejamos', 'sem', 'serei', 'seremos', 'seria', 'seriam', 'ser ',
'ser o', 'ser amos', 'seu', 'seus', 'somos', 'sou', 'sua', 'su
as', 's o', 's ', 'tamb m', 'te', 'tem', 'temos', 'tenha', 'te
nh ', 'tenhamos', 'tenho', 'terei', 'teremos', 'teria', 'teri
am', 'ter ', 'ter o', 'ter amos', 'teu', 'teus', 'teve', 'tinh
a', 'tinham', 'tive', 'tivemos', 'tiver', 'tivera', 'tiveram',
'tiverem', 'tivermos', 'tivesse', 'tivessem', 'tiv ramos', 'ti
v ssimos', 'tu', 'tua', 'tuas', 't m', 't nhados', 'um', 'um
a', 'v c ', 'voc s', 'vos', ' ', ' s', ' ', ' ramos']
```

Instala a biblioteca wordcloud e importa as bibliotecas necess rias

In [9]:

```
# Caso j  esteja instalada atualiza
!pip install wordcloud -q
```

In [10]:

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

Cria a string contendo todas as descri es

In [11]:

```
# Mostra algumas descrições do dataset
df['DESCRICA0 DO PRODUTO'].sample(5)
```

Out[11]:

```
3427    ITEM 100240WR - KIT TRANSMISSÃO PARA MOTOCICLE...
17013    KIT DE TRANSMISSAO, MARCA RIFFEL, TITANIUM (10...
13289    KIT DE TRANSMISSAO PARA MOTOCICLETAS MODELO: N...
5509     KIT DE TRANSMISSAO , MARCA RIFFEL, TITANIUM (1...
3212     71788 - KIT TRANSMISSAO TITANIUM PARA MOTOCICL...
Name: DESCRICA0 DO PRODUTO, dtype: object
```

In [12]:

```
# Mescla todas as descrições como uma string usando espaço como separador
descricaoes = " ".join(df['DESCRICA0 DO PRODUTO']).lower()
```

Gera a WordCloud aplicando o filtro das stopwords

In [13]:

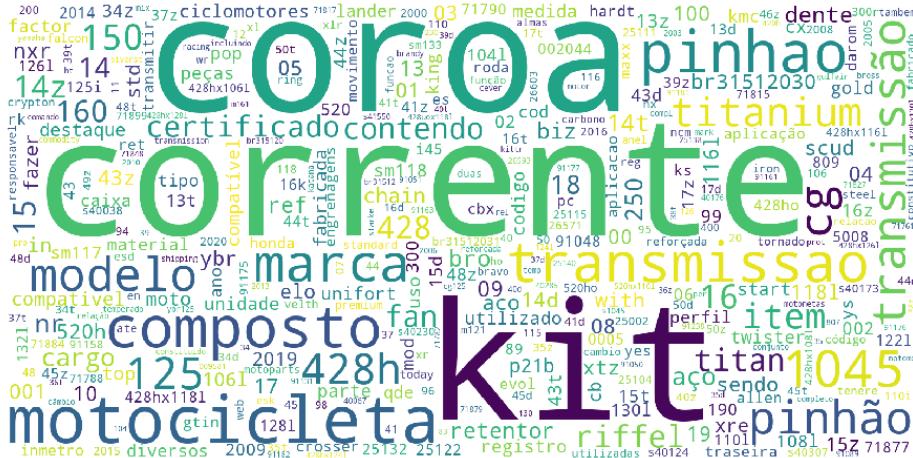
```
# Define e gera a wordcloud para um máximo de 400 palavras de tamanho mínimo 2, sem termos duplos
wordcloud = WordCloud(stopwords=stopwords,
                      background_color="white",
                      width=1600, height=800,
                      max_words=400,
                      min_word_length=2,
                      collocations=False,
                      include_numbers=True).generate(descricaoes)
```

In [14]:

```
# Exibe a imagem da WordCloud gerada
fig, ax = plt.subplots(figsize=(20,8))
ax.imshow(wordcloud, interpolation='bilinear')
ax.set_axis_off()
plt.imshow(wordcloud)
```

Out[14]:

<matplotlib.image.AxesImage at 0x165c41c65f8>



In [15]:

```
# Exporta para um arquivo
wordcloud.to_file(r"./imagens/wordcloud_descricoes_anteriores.png")
```

Out[15]:

<wordcloud.wordcloud.WordCloud at 0x165c41ba748>

Atualiza as stopwords contendo as palavras irrelevantes

Palavras a adicionar, tais como: kit, transmissao, transmissão, coroa, pinhão, etc. que não agregam nenhuma diferença aos itens da lista

In [16]:

```
# 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']))
swtemp = list(stopwords_df)
swtemp.sort()
print(swtemp)
```

['abaixo', 'acessorios', 'acessórios', 'aco', 'acondicionado s', 'adaptavel', 'adaptável', 'allen', 'almas', 'alta', 'am', 'anel', 'ano', 'aplicacao', 'application', 'aplicavel', 'aplica ção', 'aplicável', 'application', 'ate', 'atitanium', 'aç', 'a ço', 'bicicleta', 'bicycle', 'bike', 'bravo', 'cada', 'caixa', 'caixas', 'cambio', 'carbono', 'certificado', 'cever', 'chai n', 'chh', 'china', 'ciclomotor', 'ciclomotores', 'cilindrad a', 'cilindradas', 'cod', 'code', 'codigo', 'comando', 'combus tão', 'comercial', 'comercialmente', 'commodity', 'compativel', 'compatível', 'compl', 'completo', 'completos', 'compost o', 'composto', 'compostopor', 'compostpo', 'comum', 'condicao', 'condicoes', 'condição', 'condições', 'confeccionado', 'co nformidade', 'conhecido', 'conj', 'conjunto', 'conjuntos', 'co nstituido', 'constitutivo', 'constituído', 'contendo', 'copost o', 'coroa', 'coroaes', 'corr', 'current', 'corrente', 'corren tee', 'correntes', 'corrnte', 'cx', 'câmbio', 'câmbio', 'código', 'decreto', 'denominada', 'dente', 'dentes', 'descricao', 'descricão', 'descrição', 'destaque', 'destaque', 'destaques', 'detransmissão', 'diante', 'dimensao', 'dimensoes', 'dimensão', 'dimensões', 'diverso', 'diversos', 'dominad o', 'durabilidade', 'elo', 'elos', 'embalagem', 'engine', 'eng renagem', 'engrenagens', 'epinhao', 'epinhão', 'especifico', 'específico', 'espessura', 'evol', 'exclusivo', 'fabr', 'fabri', 'fabricada', 'fabricado', 'final', 'foiproduzida', 'formad o', 'funcao', 'funcao', 'funcão', 'funcão', 'funçao', 'funçã o', 'gtin', 'hardt', 'ho', 'hp', 'imetro', 'in', 'incluindo', 'incluso', 'indicado', 'ingles', 'inmetro', 'inv', 'invoice', 'iron', 'item', 'jc', 'ki', 'kif', 'kit', 'kitr', 'kittr', 'kittr', 'kmc', 'ligacoes', 'ligações', 'ligações', 'marca', 'mark', 'match', 'material', 'materialdo', 'maxx', 'medida', 'medidas', 'medindo', 'metal', 'mini', 'mod', 'modelo', 'modelos', 'moto', 'motociclet', 'motocicleta', 'motocicle tas', 'motoneta', 'motonetas', 'motoparts', 'motor', 'motorcic leta', 'motos', 'motos', 'movimento', 'nbsp', 'ncm', 'nome', 'nopinh', 'normais', 'nova', 'novo', 'nr', 'numero', 'número', 'onde', 'or', 'origem', 'oring', 'ox', 'papelao', 'papelão', 'part', 'parte', 'partes', 'parts', 'pc', 'pc-coroa', 'pc-correcte', 'pc-pinhao', 'pcs', 'pec', 'pecas', 'perfil', 'peç', 'peças', 'pinh', 'pinhao', 'pinhão', 'posição', 'premium', 'proc edencia', 'procedência', 'prodepe', 'produto', 'próprio', 'p ç', 'qdes', 'qtd', 'qtds', 'qty', 'quadriciclo', 'quadriciclos', 'quantidad', 'quantidade', 're', 'ref', 'reforcada', 'refo rçada', 'registro', 'rel', 'relacao', 'relação', 'reposicao', 'resp', 'respo', 'respons', 'responsa', 'responsav', 'responsa ve', 'responsavel', 'responsáv', 'responsável', 'ret', 'retalh o', 'retentor', 'riffel', 'ring', 'roda', 'sae', 'scud', 'semi', 'semi-', 'semi-kit', 'sendo', 'serve', 'set', 'shipping', 'sistema', 'sm', 'sprocket', 'standard', 'standart', 'standart t', 'std', 'stdmodelo', 'steel', 'tambem', 'também', 'tec', 'temp', 'temperado', 'tipo', 'tipos', 'titani', 'titaniu', 'tit anium', 'titanium', 'tr', 'tracao', 'tracão', 'trans', 'transmis', 'transmisaõ', 'transmiss', 'transmissa', 'transmissao', 'transmission', 'transmissão', 'transmitir', 'traseira', 'traçao', 'tração', 'und', 'unds', 'unid', 'unidade', 'unidade', 'unidades', 'unifort', 'uo', 'uso', 'utilizada', 'utilizadas',

```
'utilizado', 'utilizados', 'utilização', 'vem', 'venda', 'wit  
h', 'xy', 'year']
```

In [17]:

```
# Atualizar stopwords  
stopwords.update(stopwords_df)
```

In [18]:

```
# Mostra toda a lista de stopwords
swtemp = list(stopwords)
swtemp.sort()
print(swtemp)
```

['a', 'abaixo', 'acessorios', 'acessórios', 'aco', 'acondicionados', 'adaptavel', 'adaptável', 'allen', 'almas', 'alta', 'am', 'anel', 'ano', 'ao', 'aos', 'aplicacao', 'application', 'aplicavel', 'aplicação', 'aplicável', 'application', 'aquela', 'aqueelas', 'aquele', 'aqueles', 'aquito', 'as', 'ate', 'atitanium', 'até', 'ac', 'aço', 'bicicleta', 'bicycle', 'bike', 'bravo', 'cada', 'caixa', 'caixas', 'cambio', 'carbono', 'certificado', 'cever', 'chain', 'chh', 'china', 'ciclomotor', 'ciclomotores', 'cilindrada', 'cilindradas', 'cod', 'code', 'codigo', 'com', 'comando', 'combustão', 'comercial', 'comercialmente', 'commodity', 'como', 'compativel', 'compatível', 'compl', 'completo', 'completos', 'composto', 'compostopor', 'compostpo', 'comum', 'condicao', 'condicoes', 'condição', 'condições', 'confecionado', 'conformidade', 'conhecido', 'conj', 'conjunto', 'conjuntos', 'constituido', 'constitutivo', 'constituído', 'contendo', 'coposto', 'coroa', 'coroaes', 'corr', 'corrent', 'corrente', 'correntee', 'correntes', 'corrnte', 'cx', 'câmbio', 'câmbio', 'código', 'da', 'das', 'de', 'decreto', 'dela', 'delas', 'dele', 'deles', 'denominada', 'dente', 'dentes', 'depois', 'descricao', 'descrição', 'descriçao', 'descrição', 'destaque', 'destaques', 'detransmissão', 'diante', 'dimensao', 'dimensoes', 'dimensão', 'dimensões', 'diverso', 'diversos', 'do', 'dominado', 'dos', 'durabilidade', 'e', 'ela', 'elas', 'ele', 'eles', 'elo', 'elos', 'em', 'embalagem', 'engine', 'engrenage', 'engrenagens', 'entre', 'epinhao', 'epinhão', 'era', 'era', 'especifico', 'específico', 'espessura', 'essa', 'essas', 'esse', 'esses', 'esta', 'estamos', 'estas', 'estava', 'estavam', 'este', 'esteja', 'estejam', 'estejamos', 'estes', 'esteve', 'estive', 'estivemos', 'estiver', 'estivera', 'estiveram', 'estiverem', 'estivermos', 'estivesse', 'estivessem', 'estivéramos', 'estivéssemos', 'estou', 'está', 'estávamos', 'estão', 'eu', 'evol', 'exclusivo', 'fabr', 'fabri', 'fabricada', 'fabricado', 'final', 'foi', 'foiproduzida', 'fomos', 'for', 'fora', 'foram', 'forem', 'formado', 'formos', 'fosse', 'fossem', 'fui', 'funcao', 'funcão', 'função', 'fôramos', 'fôssemos', 'gtin', 'haja', 'hajam', 'hajamos', 'hardt', 'havemos', 'hei', 'ho', 'houve', 'houvemos', 'houver', 'houvera', 'houveram', 'houverei', 'houverem', 'houveremos', 'houveria', 'houveriam', 'houvermos', 'houverá', 'houverão', 'houveríamos', 'houvesse', 'houvessem', 'houvéramos', 'houvéssemos', 'hp', 'há', 'hão', 'imetro', 'in', 'incluindo', 'incluso', 'indicado', 'ingles', 'inmetro', 'inv', 'invoice', 'iron', 'isso', 'isto', 'item', 'jc', 'já', 'ki', 'kif', 'kit', 'kitr', 'kittr', 'km', 'lhe', 'lhes', 'ligacoes', 'ligações', 'ligações', 'mais', 'marca', 'mark', 'mas', 'match', 'material', 'materialdo', 'maxx', 'me', 'medida', 'medidas', 'medindo', 'mesmo', 'metal', 'meu', 'meus', 'minha', 'minhas', 'mini', 'mod', 'modelo', 'modelos', 'moto', 'motociclet', 'motocicleta', 'motocicletas', 'motoneta', 'motonetas', 'motoparts', 'motor', 'motorcicleta', 'motos', 'movimento', 'muito', 'na', 'nas', 'nbsp', 'ncm', 'nem', 'no', 'nome', 'nopinh', 'normais', 'nos', 'nossa', 'nossas', 'nosso', 'nossos', 'nova', 'novo', 'nr', 'num', 'numa', 'numero', 'não', 'nós', 'número', 'o', 'onde', 'or', 'origem', 'oring', 'os', 'ou', 'ox', 'papelao', 'papelão', 'para', 'part', 'parte', 'partes', 'parts', 'pc', 'pc-coroa', 'pc-corrente', 'pc-pinhao', 'pcs', 'pec', 'pecas', 'pela', 'pelas', 'pel

o', 'pelos', 'perfil', 'peç', 'peças', 'pinh', 'pinhao', 'pinhão', 'por', 'posição', 'premium', 'procedencia', 'procedênci a', 'prodepe', 'produto', 'próprio', 'pç', 'qdes', 'qtd', 'qtd s', 'qty', 'quadriciclo', 'quadriciclos', 'qual', 'quando', 'q uantidad', 'quantidade', 'que', 'quem', 're', 'ref', 'reforcad a', 'reforçada', 'registro', 'rel', 'relacao', 'relação', 'rep osicao', 'resp', 'respo', 'respons', 'responsa', 'responsav', 'responsave', 'responsavel', 'responsável', 're t', 'retalho', 'retentor', 'riffel', 'ring', 'roda', 'sae', 's cud', 'se', 'seja', 'sejam', 'sejamos', 'sem', 'semi', 'semi- ', 'semi-kit', 'sendo', 'serei', 'seremos', 'seria', 'seriam', 'serve', 'será', 'serão', 'seríamos', 'set', 'seu', 'seus', 's hipping', 'sistema', 'sm', 'somos', 'sou', 'sprocket', 'standa rd', 'standart', 'standartt', 'std', 'stdmodelo', 'steel', 'su a', 'suas', 'são', 'só', 'tambem', 'também', 'te', 'tec', 'te m', 'temos', 'temp', 'temperado', 'tenha', 'tenham', 'tenhamo s', 'tenho', 'terei', 'teremos', 'teria', 'teriam', 'terá', 'terão', 'teríamos', 'teu', 'teus', 'teve', 'tinha', 'tinham', 'tipo', 'tipos', 'titanio', 'titaniu', 'titanium', 'titanium', 'tive', 'tivemos', 'tiver', 'tivera', 'tiveram', 'tiverem', 't ivemos', 'tivesse', 'tivessem', 'tivéramos', 'tivéssemos', 't r', 'tracao', 'tracão', 'trans', 'transmis', 'transmisao', 'transmiss', 'transmissa', 'transmissao', 'transmission', 'transm issão', 'transmitir', 'traseira', 'traçao', 'tração', 'tu', 't ua', 'tuas', 'tém', 'tínhamos', 'um', 'uma', 'und', 'unds', 'u nid', 'unidade', 'unidades', 'unifort', 'uo', 'uso', 'utilizad a', 'utilizadas', 'utilizado', 'utilizados', 'utilizaçao', 've m', 'venda', 'você', 'vocês', 'vos', 'with', 'xy', 'year', 'à', 'às', 'é', 'éramos']

Gera a WodCloud aplicando o filtro das stopwords atualizado

In [19]:

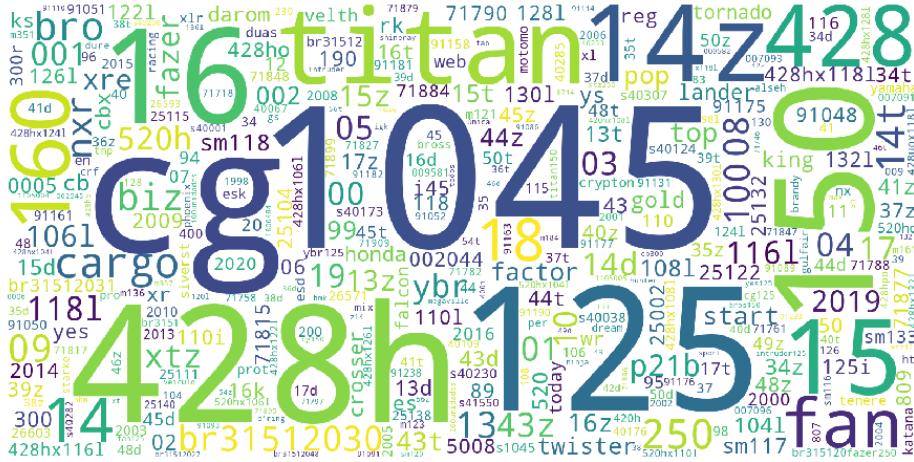
```
# Define e gera a wordcloud para um máximo de 400 palavras de tamanho mínimo 2, sem termos duplos
wordcloud = WordCloud(stopwords=stopwords,
                      background_color="white",
                      width=1600, height=800,
                      max_words=400,
                      min_word_length=2,
                      collocations=False,
                      include_numbers=True).generate(descricoes)
```

In [20]:

```
# Exibe a imagem da nova WordCloud gerada
fig, ax = plt.subplots(figsize=(20,8))
ax.imshow(wordcloud, interpolation='bilinear')
ax.set_axis_off()
plt.imshow(wordcloud)
```

Out[20]:

<matplotlib.image.AxesImage at 0x165c42132b0>



In [21]:

```
# Exporta para um arquivo
wordcloud.to_file("./imagens/wordcloud_descricoes_depois.png")
```

Out[21]:

<wordcloud.wordcloud.WordCloud at 0x165c4208fd0>

In [22]:

```
tempotot=time.time()-initot
if tempotot>60:
    print(f'Tempo total de execução: {tempotot/60:.2f} minutos.')
else:
    print(f'Tempo total de execução: {tempotot:.2f} segundos.')
```

Tempo total de execução: 15.40 segundos.

Notebook Jupyter 3_classificarAplicação

Classificação dos modelos de motocicleta a partir da descrição

A grande dificuldade na tarefa de análise de valores compatíveis na importação de peças de motocicletas, em especial dos kits de transmissão, se dá no fato de que milhares de importadores adquirem essas peças no exterior e informam sua descrição em um campo texto livre.

Nem mesmo a utilização da classificação fiscal normatizada no Mercosul, chamada de Nomenclatura Comum do Mercosul – NCM, ajuda nesse caso específico, tendo em vista que grande parte das peças de motocicletas e todos os kits de transmissão são classificados em uma mesma posição na tabela da NCM.

Para que se possa tratar corretamente o dataset obtido na nossa etapa de processamento e tratamento de dados, e permitir o futuro aprendizado de máquina, com predições do modelo de motocicleta que aquele item se aplica, é preciso que primeiro se proceda a uma classificação de aplicações que futuramente será utilizado em um aprendizado supervisionado.

A ideia é se aplicar uma busca na descrição da mercadoria pelos termos conhecidos de aplicações e se buscar a qual aplicação aquela descrição se refere, fazendo desse modo a primeira classificação.

Posteriormente será utilizado um algoritmo de aprendizado de máquina para aprender com o próprio texto da descrição da aplicação e fazer a classificação utilizando a descrição já limpa de stopwords e outros termos desnecessários.

A interseção dos dois conjuntos de classificação será o dataset utilizado para fazer o treinamento do classificador, que será o primeiro passo antes da análise de valor do item importado.

Importa as bibliotecas necessárias

In [1]:

```
import pandas as pd, numpy as np
import re, time
# stopwords
from nltk.corpus import stopwords
# wordCloud
from wordcloud import WordCloud
# plotagem do gráfico
import matplotlib.pyplot as plt
```

In [2]:

```
# Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
```

Código executado em 20/01/2022 às 16:36

Importa os dados já tratados

In [3]:

```
# Importa base de dados para um dataframe
df = pd.read_excel(r'./bases/dataframe.xlsx')
```

In [4]:

```
# Verifica o tamnanho do dataframe
df.shape
```

Out[4]:

(18276, 3)

In [5]:

```
# Mostra Linhas de exemplo do dataframe
df.sample(5)
```

Out[5]:

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR
5908	CHINA, REPUBLICA POP	10530002 IN KIT TRANSMISSAO P/MOTOCICLETAS(COR...)	3.720000
12613	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO PARA MOTOCICLETAS, COMPOSTO...	3.360000
12836	CHINA, REPUBLICA POP	881606 - KIT DE TRANSMISSAO, COMPOSTO DE CORRE...	5.024464
14707	CHINA, REPUBLICA POP	152624 # KIT TRANSMISSAO STANDARD TEMP. COMPL....	3.197229
2151	CHINA, REPUBLICA POP	71848 - KIT NXR 150 BROS ESD (03- 05) 50Z X 17Z...	5.757700

In [6]:

```
df['DESCRICAOPRODUTO'][5]
```

Out[6]:

```
'80348 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE, COROA E PINHÃ  
O PARA MOTOCICLETA CBX 250 TWISTER, MARCA ALLEN.'
```

In [7]:

```
type(df['DESCRICAOPRODUTO'][1])
```

Out[7]:

```
str
```

Importa as stopwords da língua portuguesa

In [8]:

```
# Importar lista de Stopwords  
stopwords = set(stopwords.words('portuguese'))
```

In [9]:

```
# Mostra tamanho da Lista de stopwords  
len(stopwords)
```

Out[9]:

```
204
```

In [10]:

```
# Mostra toda a lista de stopwords
swtemp = list(stopwords)
swtemp.sort()
print(swtemp)
```

```
['a', 'ao', 'aos', 'aqueла', 'aqueлas', 'aqueле', 'aqueлes',
'aquilo', 'as', 'at ', 'com', 'como', 'da', 'das', 'de', 'del
a', 'delas', 'dele', 'deles', 'depois', 'do', 'dos', 'e', 'el
a', 'elas', 'ele', 'eles', 'em', 'entre', 'era', 'eram', 'ess
a', 'essas', 'esse', 'esses', 'esta', 'estamos', 'estas', 'est
ava', 'estavam', 'este', 'esteja', 'estejam', 'estejamos', 'es
tes', 'esteve', 'estive', 'estivemos', 'estiver', 'estivera',
'estiveram', 'estiverem', 'estivermos', 'estivesse', 'estivess
em', 'estiv ramos', 'estiv ssimos', 'estou', 'est ', 'est vamo
s', 'est o', 'eu', 'foi', 'fomos', 'for', 'fora', 'foram', 'fo
rem', 'formos', 'fosse', 'fossem', 'fui', 'f ramos', 'f ssimo
s', 'haja', 'hajam', 'hajamos', 'havemos', 'hei', 'houve', 'ho
uvemos', 'houver', 'houvera', 'houveram', 'houverei', 'houvere
m', 'houveremos', 'houveria', 'houveriam', 'houvermos', 'houve
r ', 'houver o', 'houver amos', 'houvesse', 'houvessem', 'houv
 ramos', 'houv ssimos', 'h ', 'h o', 'isso', 'isto', 'j ', 'lh
e', 'lhes', 'mais', 'mas', 'me', 'mesmo', 'meu', 'meus', 'minh
a', 'minhas', 'muito', 'na', 'nas', 'nem', 'no', 'nos', 'noss
a', 'nossas', 'noso', 'nossos', 'num', 'numa', 'n o', 'n s',
'o', 'os', 'ou', 'para', 'pela', 'pelas', 'pelo', 'pelos', 'po
r', 'qual', 'quando', 'que', 'quem', 'se', 'seja', 'sejam', 's
ejamos', 'sem', 'serei', 'seremos', 'seria', 'seriam', 'ser ',
'ser o', 'ser amos', 'seu', 'seus', 'somos', 'sou', 'sua', 'su
as', 's o', 's o', 'tamb m', 'te', 'tem', 'temos', 'tenha', 'te
nh ', 'tenhamos', 'tenho', 'terei', 'teremos', 'teria', 'teri
am', 'ter ', 'ter o', 'ter amos', 'teu', 'teus', 'teve', 'tinh
a', 'tinham', 'tive', 'tivemos', 'tiver', 'tivera', 'tiveram',
'tiverem', 'tivermos', 'tivesse', 'tivessem', 'tiv ramos', 'ti
v ssimos', 'tu', 'tua', 'tuas', 't m', 't nhados', 'um', 'um
a', 'voc ', 'voc s', 'vos', ' ', ' s', ' ', ' ramos']
```

Atualiza a lista de stopwords

In [11]:

```
# 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']))
swtemp = list(stopwords_df)
swtemp.sort()
print(swtemp)
```

['abaixo', 'acessorios', 'acessórios', 'aco', 'acondicionado s', 'adaptavel', 'adaptável', 'allen', 'almas', 'alta', 'am', 'anel', 'ano', 'aplicacao', 'application', 'aplicavel', 'aplica ção', 'aplicável', 'application', 'ate', 'atitanium', 'aç', 'a ço', 'bicicleta', 'bicycle', 'bike', 'bravo', 'cada', 'caixa', 'caixas', 'cambio', 'carbono', 'certificado', 'cever', 'chai n', 'chh', 'china', 'ciclomotor', 'ciclomotores', 'cilindrad a', 'cilindradas', 'cod', 'code', 'codigo', 'comando', 'combus tão', 'comercial', 'comercialmente', 'commodity', 'compativel', 'compatível', 'compl', 'completo', 'completos', 'compost o', 'composto', 'compostopor', 'compostpo', 'comum', 'condicao', 'condicoes', 'condição', 'condições', 'confeccionado', 'co nformidade', 'conhecido', 'conj', 'conjunto', 'conjuntos', 'co nstituido', 'constitutivo', 'constituído', 'contendo', 'copost o', 'coroa', 'coroaes', 'corr', 'current', 'corrente', 'corren tee', 'correntes', 'corrnte', 'cx', 'câmbio', 'câmbio', 'código', 'decreto', 'denominada', 'dente', 'dentes', 'descricao', 'descricão', 'descrição', 'destaque', 'destaque', 'destaques', 'detransmissão', 'diante', 'dimensao', 'dimensoes', 'dimensão', 'dimensões', 'diverso', 'diversos', 'dominad o', 'durabilidade', 'elo', 'elos', 'embalagem', 'engine', 'eng renagem', 'engrenagens', 'epinhao', 'epinhão', 'especifico', 'específico', 'espessura', 'evol', 'exclusivo', 'fabr', 'fabri', 'fabricada', 'fabricado', 'final', 'foiproduzida', 'formad o', 'funcao', 'funcao', 'funcão', 'funcão', 'funçao', 'funçã o', 'gtin', 'hardt', 'ho', 'hp', 'imetro', 'in', 'incluindo', 'incluso', 'indicado', 'ingles', 'inmetro', 'inv', 'invoice', 'iron', 'item', 'jc', 'ki', 'kif', 'kit', 'kitr', 'kittr', 'kittr', 'kmc', 'ligacoes', 'ligações', 'ligações', 'marca', 'mark', 'match', 'material', 'materialdo', 'maxx', 'medida', 'medidas', 'medindo', 'metal', 'mini', 'mod', 'modelo', 'modelos', 'moto', 'motociclet', 'motocicleta', 'motocicle tas', 'motoneta', 'motonetas', 'motoparts', 'motor', 'motorcic leta', 'motos', 'motos', 'movimento', 'nbsp', 'ncm', 'nome', 'nopinh', 'normais', 'nova', 'novo', 'nr', 'numero', 'número', 'onde', 'or', 'origem', 'oring', 'ox', 'papelao', 'papelão', 'part', 'parte', 'partes', 'parts', 'pc', 'pc-coroa', 'pc-correcte', 'pc-pinhao', 'pcs', 'pec', 'pecas', 'perfil', 'peç', 'peças', 'pinh', 'pinhao', 'pinhão', 'posição', 'premium', 'proc edencia', 'procedência', 'prodepe', 'produto', 'próprio', 'p ç', 'qdes', 'qtd', 'qtds', 'qty', 'quadriciclo', 'quadriciclos', 'quantidad', 'quantidade', 're', 'ref', 'reforcada', 'refo rçada', 'registro', 'rel', 'relacao', 'relação', 'reposicao', 'resp', 'respo', 'respons', 'responsa', 'responsav', 'responsa ve', 'responsavel', 'responsáv', 'responsável', 'ret', 'retalh o', 'retentor', 'riffel', 'ring', 'roda', 'sae', 'scud', 'semi', 'semi-', 'semi-kit', 'sendo', 'serve', 'set', 'shipping', 'sistema', 'sm', 'sprocket', 'standard', 'standart', 'standart t', 'std', 'stdmodelo', 'steel', 'tambem', 'também', 'tec', 'temp', 'temperado', 'tipo', 'tipos', 'titani', 'titaniu', 'tit anium', 'titanium', 'tr', 'tracao', 'tracão', 'trans', 'transmis', 'transmisaõ', 'transmiss', 'transmissa', 'transmissao', 'transmission', 'transmissão', 'transmitir', 'traseira', 'traçao', 'tração', 'und', 'unds', 'unid', 'unidade', 'unidade', 'unidades', 'unifort', 'uo', 'uso', 'utilizada', 'utilizadas',

```
'utilizado', 'utilizados', 'utilização', 'vem', 'venda', 'wit  
h', 'xy', 'year']
```

In [12]:

```
# Atualizar stopwords  
stopwords.update(stopwords_df)
```

In [13]:

```
# Mostra toda a lista de stopwords
swtemp = list(stopwords)
swtemp.sort()
print(swtemp)
```

['a', 'abaixo', 'acessorios', 'acessórios', 'aco', 'acondicionados', 'adaptavel', 'adaptável', 'allen', 'almas', 'alta', 'am', 'anel', 'ano', 'ao', 'aos', 'aplicacao', 'application', 'aplicavel', 'aplicação', 'aplicável', 'application', 'aquela', 'aqueelas', 'aquele', 'aqueles', 'aquilo', 'as', 'ate', 'atitanium', 'até', 'ac', 'aço', 'bicicleta', 'bicycle', 'bike', 'bravo', 'cada', 'caixa', 'caixas', 'cambio', 'carbono', 'certificado', 'cever', 'chain', 'chh', 'china', 'ciclomotor', 'ciclomotores', 'cilindrada', 'cilindradas', 'cod', 'code', 'codigo', 'com', 'comando', 'combustão', 'comercial', 'comercialmente', 'commodity', 'como', 'compativel', 'compatível', 'compl', 'completo', 'completos', 'composto', 'compostopor', 'compostpo', 'comum', 'condicao', 'condicoes', 'condição', 'condições', 'confeccionado', 'conformidade', 'conhecido', 'conj', 'conjunto', 'conjuntos', 'constituido', 'constitutivo', 'constituído', 'contendo', 'coposto', 'coroa', 'coroaes', 'corr', 'corrent', 'corrente', 'correntee', 'correntes', 'corrnte', 'cx', 'câmbio', 'câmbio', 'código', 'da', 'das', 'de', 'decreto', 'dela', 'delas', 'dele', 'deles', 'denominada', 'dente', 'dentes', 'depois', 'descricao', 'descrição', 'descriçao', 'descrição', 'destaque', 'destaques', 'detransmissão', 'diante', 'dimensao', 'dimensoes', 'dimensão', 'dimensões', 'diverso', 'diversos', 'do', 'dominado', 'dos', 'durabilidade', 'e', 'ela', 'elas', 'ele', 'eles', 'elo', 'elos', 'em', 'embalagem', 'engine', 'engrenage', 'engrenagens', 'entre', 'epinhao', 'epinhão', 'era', 'era', 'especifico', 'específico', 'espessura', 'essa', 'essas', 'esse', 'esses', 'esta', 'estamos', 'estas', 'estava', 'estavam', 'este', 'esteja', 'estejam', 'estejamos', 'estes', 'esteve', 'estive', 'estivemos', 'estiver', 'estivera', 'estiveram', 'estiverem', 'estivermos', 'estivesse', 'estivessem', 'estivéramos', 'estivéssemos', 'estou', 'está', 'estávamos', 'estão', 'eu', 'evol', 'exclusivo', 'fabr', 'fabri', 'fabricada', 'fabricado', 'final', 'foi', 'foiproduzida', 'fomos', 'for', 'fora', 'foram', 'forem', 'formado', 'formos', 'fosse', 'fossem', 'fui', 'funcao', 'funcão', 'função', 'fôramos', 'fôssemos', 'gtin', 'haja', 'hajam', 'hajamos', 'hardt', 'havemos', 'hei', 'ho', 'houve', 'houvemos', 'houver', 'houvera', 'houveram', 'houverei', 'houverem', 'houveremos', 'houveria', 'houveriam', 'houvermos', 'houverá', 'houverão', 'houveríamos', 'houvesse', 'houvessem', 'houvéramos', 'houvéssemos', 'hp', 'há', 'hão', 'imetro', 'in', 'incluindo', 'incluso', 'indicado', 'ingles', 'inmetro', 'inv', 'invoice', 'iron', 'isso', 'isto', 'item', 'jc', 'já', 'ki', 'kif', 'kit', 'kitr', 'kittr', 'km', 'lhe', 'lhes', 'ligacoes', 'ligações', 'ligações', 'mais', 'marca', 'mark', 'mas', 'match', 'material', 'materialdo', 'maxx', 'me', 'medida', 'medidas', 'medindo', 'mesmo', 'metal', 'meu', 'meus', 'minha', 'minhas', 'mini', 'mod', 'modelo', 'modelos', 'moto', 'motociclet', 'motocicleta', 'motocicletas', 'motoneta', 'motonetas', 'motoparts', 'motor', 'motorcicleta', 'motos', 'movimento', 'muito', 'na', 'nas', 'nbsp', 'ncm', 'nem', 'no', 'nome', 'nopinh', 'normais', 'nos', 'nossa', 'nossas', 'nosso', 'nossos', 'nova', 'novo', 'nr', 'num', 'numa', 'numero', 'não', 'nós', 'número', 'o', 'onde', 'or', 'origem', 'oring', 'os', 'ou', 'ox', 'papelao', 'papelão', 'para', 'part', 'parte', 'partes', 'parts', 'pc', 'pc-coroa', 'pc-corrente', 'pc-pinhao', 'pcs', 'pec', 'pecas', 'pela', 'pelas', 'pel'

```

o', 'pelos', 'perfil', 'peç', 'peças', 'pinh', 'pinhao', 'pinh
ão', 'por', 'posição', 'premium', 'procedencia', 'procedênci
a', 'prodepe', 'produto', 'próprio', 'pç', 'qdes', 'qtd', 'qtd
s', 'qty', 'quadriciclo', 'quadriciclos', 'qual', 'quando', 'q
uantidad', 'quantidade', 'que', 'quem', 're', 'ref', 'reforcad
a', 'reforçada', 'registro', 'rel', 'relacao', 'relação', 'rep
osicao', 'resp', 'respo', 'respons', 'responsa', 'responsav',
'responsave', 'responsavel', 'responsáv', 'responsável', 're
t', 'retalho', 'retentor', 'riffel', 'ring', 'roda', 'sae', 's
cud', 'se', 'seja', 'sejam', 'sejamos', 'sem', 'semi', 'semi-
', 'semi-kit', 'sendo', 'serei', 'seremos', 'seria', 'seriam',
'serve', 'será', 'serão', 'seríamos', 'set', 'seu', 'seus', 's
hipping', 'sistema', 'sm', 'somos', 'sou', 'sprocket', 'standa
rd', 'standart', 'standartt', 'std', 'stdmodelo', 'steel', 'su
a', 'suas', 'são', 'só', 'tambem', 'também', 'te', 'tec', 'te
m', 'temos', 'temp', 'temperado', 'tenha', 'tenham', 'tenhamo
s', 'tenho', 'terei', 'teremos', 'teria', 'teriam', 'terá', 't
erão', 'teríamos', 'teu', 'teus', 'teve', 'tinha', 'tinham',
'tipo', 'tipos', 'titania', 'titaniu', 'titanium', 'titanium',
'tive', 'tivemos', 'tiver', 'tivera', 'tiveram', 'tiverem', 't
ivermos', 'tivesse', 'tivessem', 'tivéramos', 'tivéssemos', 't
r', 'tracao', 'tracão', 'trans', 'transmis', 'transmisao', 'tr
ansmiss', 'transmissa', 'transmissao', 'transmission', 'transm
issão', 'transmitir', 'traseira', 'traçao', 'tração', 'tu', 't
ua', 'tuas', 'tém', 'tínhamos', 'um', 'uma', 'und', 'unds', 'u
nid', 'unidade', 'unidades', 'unifort', 'uo', 'uso', 'utilizad
a', 'utilizadas', 'utilizado', 'utilizados', 'utilização', 've
m', 'venda', 'você', 'vocês', 'vos', 'with', 'xy', 'year',
'à', 'às', 'é', 'éramos']

```

In [14]:

```
len(stopwords)
```

Out[14]:

518

Carrega lista de aplicações

In [15]:

```
# carrega a lista de marcas de motos do arquivo
dftemp=pd.read_csv('./bases/Aplicacoes.csv')
```

In [16]:

```
dftemp.head()
```

Out[16]:

APLICACOES	
0	ACELLERA ACX 250F 250
1	ACELLERA FRONTLANDER 500
2	ACELLERA FRONTLANDER 800 EFI
3	ACELLERA HOTZOO SPORT 90
4	ACELLERA QUADRILANDER 300

Cria a lista de Palavras Chave das Aplicações

In [17]:

```
# remove caracteres especiais ou soltos e termos duplicados, salvando na lista
palavrasChave=sorted(set(re.sub(r"\b \w \b",
                                '',
                                re.sub(r"/<>()|+\$%#@\'\"]+",',
                                '',
                                " ".join(dftemp['APLICACOES'].tolist
                                         ))).split()))
```

In [18]:

```
len(palavrasChave)
```

Out[18]:

894

In [19]:

```
# amostra de palavrasChave
print(palavrasChave[:20], ' ... ', palavrasChave[-20:])
```

```
['1000', '1000F', '1000R', '1000V', '1098', '1100', '1100XX',
'110S', '1125', '1190', '1198', '1200', '1200Z', '125', '125
R', '130', '1300', '135', '1400', '150'] ... ['YFS', 'YS15
0', 'YS250', 'YZ', 'YZF', 'YZR', 'Z1000', 'Z750', 'Z800', 'ZAC
H', 'ZANELLA', 'ZENITH', 'ZIG', 'ZING', 'ZIP', 'ZONGSHEN', 'Z
R', 'ZRX', 'ZS', 'ZZR']
```

In [20]:

```
'RT' in palavrasChave
```

Out[20]:

True

Limpeza e criação da coluna DESCRIAO

Função de limpeza de dados irrelevantes para a classificação e remoção de stopwords

In [21]:

```

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"([^\w\s])\b(ho|uo|h|l|t|kt\d|sm|m|d|x|elos)\d{1,}[ \-\/,);.]/[ \x\-\/\(\]*\d{1,}(ho|uo|h|l|z|t|kt\d|m|d|x| dentes| elos)[ \-\/,;]", ' ', descricao) # 00h
    descricao=re.sub(r"\d*(ho|uo|h|l|t|kt\d|sm|m|d|elos)\d{1,}[ \-\/,);.]/[ \-\/(\]*\d{1,}(ho|uo|h|l|z|t|kt\d|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','hnnda','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 caracteres 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|pc|pec|pec)( \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 espa
    ção
    for marca in ['kmc *gold','am *gold','king','bravo *racing','riffel *to
    p']:
        descricao=re.sub(r"\bmarca[ :./]*"+str(marca)+r"[^ ]*", ' ', descr
        icao) # 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 termo
        s 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)) # pe
        ga 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

```

Exemplo de execução da função

In [22]:

```
linha=745
```

In [23]:

```
df.iloc[linha]['DESCRICAOPRODUTO']
```

Out[23]:

```
'KIT TRANSMISSÃO PARA MOTOCICLETAS, COMPOSTO DE CORRENTE, PINHAO  
E COROA, PARA MODELOS DIVERSOS DE MOTOCICLETAS (KIT C 100 BIZ  
(13-15) 34Z X 14Z WITH CHAIN 428H X 108L - TITANIUM (1045)) MO  
DELO KIT C 100 BIZ (13-15) 34Z X 14Z WITH CHAIN 428H X 108L -  
TIT'
```

In [24]:

```
limpaDescricao(df.iloc[linha]['DESCRICAOPRODUTO'])
```

Out[24]:

```
'c100 biz titan'
```

Execução da função para criação da coluna DESCRIÇÃO limpa

In [25]:

```
ini=time.time()  
df['DESCRICAOPRODUTO']=df['DESCRICAOPRODUTO'].apply(limpaDescricao)  
fim=time.time()  
print(f'Tempo de execução: {fim-ini:.2} segundos.')
```

Tempo de execução: 7.8 segundos.

In [26]:

```
df.sample(5)
```

Out[26]:

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO
10292	CHINA, REPUBLICA POP	-54T17T/428H132L - KIT TRANSMISSAO EM ACO COMP...	3.600100	
1897	CHINA, REPUBLICA POP	ENGRENAGENS PARA TRANSMISSÃO DE MOTOCICLETAS E...	3.713600	c100 biz
12069	CHINA, REPUBLICA POP	10530031 IN KIT TRANSMISSAO P/MOTOCICLETAS(COR...	3.744000	spee
2927	CHINA, REPUBLICA POP	007308# KIT TRANSMISSAO STANDARD TEMP. COMPL.	3.221500	suzuki yes intruder katana 125
12202	CHINA, REPUBLICA POP	91256 KIT CG 160 TITAN (16-19) / CG 160 FAN (1...	9.711875	cg 160 titan fan start cargo

In [27]:

```
df['DESCRICAO'].iloc[linha]
```

Out[27]:

```
'c100 biz titan'
```

Criação de colunas Modelo

Função de determinação de palavras chave na coluna Modelo

In [28]:

```
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()
```

In [29]:

```
achaPalavraChave(limpaDescricao(df['DESCRICA DO PRODUTO'].iloc[linha]))
```

Out[29]:

```
'c100 titan biz'
```

Função para acrescentar a marca da motocicleta

In [30]:

```
# 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', 'FIFTY', 'PALIO', 'PGO', 'STX', 'VBLADE', 'EVO', 'HUNTER MAX'],
          'SHINERAY': ['BIKE', 'BRAVO', 'DISCOVER', 'EAGLE', 'INDIANAPOLIS', 'JET', 'NEW', 'WAVE',
                     'STRONG', 'SUPER SMART', 'VENICE', 'XY']}}
```

In [31]:

```
# 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."
```

In [32]:

```
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
```

In [33]:

```
acrescentaMarca(achaPalavraChave(limpaDescricao(df['DESCRICA0 DO PRODUTO'].iloc[linha])))
```

Out[33]:

```
'biz c100 honda titan'
```

Aplica as funções

Tenha paciência, demora cerca de 1 minuto para cada mil registros.

In [34]:

```
# cria as colunas
df=df.assign(Modelo=df['DESCRICA0'])
df.iloc[linha]['DESCRICA0']
```

Out[34]:

```
'c100 biz titan'
```

In [35]:

```
df.iloc[:, -2:].sample(5)
```

Out[35]:

	DESCRICAO	Modelo
7430	nxr-125 broz	nxr-125 broz
7152	crf 230	crf 230
4746	titan 160	titan 160
15923	xls 125 96	xls 125 96
4524	xls-125	xls-125

In [36]:

```
# aplica as funções a cada coluna
ini=time.time()
now = time.strftime("%H:%M", time.localtime(time.time()))
print("Hora de início:" + now)
print(f"Tempo estimado de execução: {df.shape[0]//1000} minutos.") # 1000 registros por minuto

print('\nBuscando palavras chave... Aguarde...')
df['Modelo']=df['Modelo'].apply(achaPalavraChave)

print('\nBuscando marcas... Aguarde...')
df['Modelo']=df['Modelo'].apply(acrescentaMarca)

now = time.strftime("%H:%M", time.localtime(time.time()))
fim=time.time()
print("\nHora de término:" + str(now))
print("Tempo decorrido: " + str(round((fim-ini)/60,2)) + " minutos.")
```

Hora de início:16:36

Tempo estimado de execução: 18 minutos.

Buscando palavras chave... Aguarde...

Buscando marcas... Aguarde...

Hora de término:16:52

Tempo decorrido: 16.57 minutos.

In [37]:

```
df['DESCRICAO DO PRODUTO'].iloc[linha]
```

Out[37]:

'KIT TRANSMISSÃO PARA MOTOCICLETAS, COMPOSTO DE CORRENTE, PINHAO E COROA, PARA MODELOS DIVERSOS DE MOTOCICLETAS (KIT C 100 BIZ (13-15) 34Z X 14Z WITH CHAIN 428H X 108L - TITANIUM (1045)) MO DELO KIT C 100 BIZ (13-15) 34Z X 14Z WITH CHAIN 428H X 108L - TIT'

In [38]:

```
df[['DESCRICAO DO PRODUTO', 'DESCRICAO', 'Modelo']].sample(5)
```

Out[38]:

	DESCRICAO DO PRODUTO	DESCRICAO	Modelo
3123	21989 - 71815 - KIT DE TRANSMISSAO PARA MOTOCI...	ybr 125 factor	125 factor yamaha ybr
13490	20257/i45 - KIT DE TRANSMISSÃO EM AÇO 1045, MA...	xtz150 crosser	150 crosser xtz
9440	TM10300 - KIT DE TRANSMISSÃO COMPOSTO DE COROA...	yamaha yes 125 14	125 suzuki yamaha yes
11269	KIT DE TRANSMISSÃO EM AÇO 1045, PARA USO EM MO...	titan	honda titan
6473	KIT DE TRANSMISSAO PARA MOTOCICLETA, CONTENDO ...	titan fan *	fan honda titan

In [39]:

```
df_sem_modelo = df[df['Modelo']=='']
df_sem_modelo['DESCRICAO'].to_excel("./bases/sem_modelo.xlsx")
```

In [40]:

```
df_sem_modelo[['DESCRICAO DO PRODUTO', 'DESCRICAO', 'Modelo']].sample(10)
```

Out[40]:

	DESCRICAO DO PRODUTO	DESCRICAO	Modelo
9362	3 - PARTES E ACESSORIOS DE MOTOCICLETA, SENDO ...		49
5586	TRANSMISSAO PARA USO EM MOTOCICLETA COMPOSTO D...		
8166	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...		
3249	45T14T/428H118L- KIT TRANSMISSAO EM ACO COMPOS...		
9048	2 - PARTES E ACESSORIOS DE MOTOCICLETA, SENDO ...	34 100	
3482	REF: 428HX118LX43TX14T (1104983) -KIT TRANSMIS...		
10253	TRANSMISSAO PARA USO EM MOTOCICLETA COMPOSTO D...		
9365	6 - PARTES E ACESSORIOS DE MOTOCICLETA, SENDO ...	45	
13083	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...		
17756	001-P21B-00600 - Kit transmissao Titanium para...		

In [41]:

```
df_sem_modelo.shape
```

Out[41]:

(792, 5)

In [42]:

```
df_sem_modelo.reset_index(drop=True)
```

Out[42]:

	PAIS DE ORIGEM	DESCRICAQ DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAQ	Modelo
0	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO, COMPOSTO DE CORAÇA, CORREN...	3.900000	f80 kits	
1	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.809000		
2	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	4.052000		
3	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.876000		
4	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.877000		
5	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.294000		
6	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.871000		
7	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	4.173000		
8	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.673000		
9	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.877000		

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO	Modelo
10	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	4.166000		
11	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.849000		
12	CHINA, REPUBLICA POP	880375 - KIT DE TRANSMISSÃO, COMPOSTO DE CORRE...	3.329221	spee	
13	CHINA, REPUBLICA POP	880349 - KIT DE TRANSMISSÃO, COMPOSTO DE CORRE...	3.813052	come	
14	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	4.166000		
15	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.809000		
16	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.213000		
17	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.279000		
18	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.871000		
19	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.871000		
20	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.909000		

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO	Modelo
21	CHINA, REPUBLICA POP	980730 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENT...	4.701278	56	
22	CHINA, REPUBLICA POP	070593 - KIT TRANSMISSÃO COMPOSTO DE CORRENTE,...	3.730000	ti x118 aço1045	
23	CHINA, REPUBLICA POP	072165 - KIT TRANSMISSÃO COMPOSTO DE CORRENTE,...	4.846000	15	
24	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	4.218000		
25	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.918000		
26	CHINA, REPUBLICA POP	KIT TRANSMISSÃO AÇO (1045), COMPOSTO DE CORREN...	3.984000		
27	CHINA, REPUBLICA POP	43T14T/428H116L - KIT TRANSMISSAO EM ACO COMPO...	3.024853		
28	CHINA, REPUBLICA POP	43T16T/428H118L - KIT TRANSMISSAO EM ACO COMPO...	2.975183		
29	CHINA, REPUBLICA POP	80422 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	9.480000	mt r3 43	
...
762	CHINA, REPUBLICA POP	001-P21B-03700 - Kit transmissao Titanium para...	4.100000		
763	CHINA, REPUBLICA POP	001-P21B-04900 - Kit transmissao Titanium para...	4.550000		
764	CHINA, REPUBLICA POP	001-P21B-06100 - Kit transmissao Titanium para...	6.240000		
765	CHINA, REPUBLICA POP	001-P21B-06500 - Kit transmissao Titanium para...	3.460000		

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO	Modelo
766	CHINA, REPUBLICA POP	001-P21B-06600 - Kit transmissao Titanium para...	3.440000		
767	CHINA, REPUBLICA POP	TRANSMISSAO PARA USO EM MOTOCICLETA COMPOSTO D...	3.990000	ft2368	
768	CHINA, REPUBLICA POP	TRANSMISSAO PARA USO EM MOTOCICLETA COMPOSTO D...	3.990000	ft2388	
769	CHINA, REPUBLICA POP	12565-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.341216		
770	CHINA, REPUBLICA POP	12570-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.341216		
771	CHINA, REPUBLICA POP	12588-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.200000		
772	CHINA, REPUBLICA POP	12591-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.359939		
773	CHINA, REPUBLICA POP	12603-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
774	CHINA, REPUBLICA POP	12604-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
775	CHINA, REPUBLICA POP	12578-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
776	CHINA, REPUBLICA POP	12580-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.600000		
777	CHINA, REPUBLICA POP	12606-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
778	CHINA, REPUBLICA POP	12566-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.600000		

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO	Modelo
779	CHINA, REPUBLICA POP	12562-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
780	CHINA, REPUBLICA POP	12582-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
781	CHINA, REPUBLICA POP	12583-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.512246		
782	CHINA, REPUBLICA POP	12584-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.512246		
783	CHINA, REPUBLICA POP	12585-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
784	CHINA, REPUBLICA POP	12586-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
785	CHINA, REPUBLICA POP	12567-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	1.946308		
786	CHINA, REPUBLICA POP	12592-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
787	CHINA, REPUBLICA POP	12593-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
788	CHINA, REPUBLICA POP	12594-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	3.200000		
789	CHINA, REPUBLICA POP	12595-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.260000		
790	CHINA, REPUBLICA POP	12596-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.260000		
791	CHINA, REPUBLICA POP	12600-E - PARTES E PEÇAS DE MOTOCICLETAS, SEND...	2.260000		

792 rows × 5 columns

In [43]:

```
print(f'Número de registros sem aplicação contida na descrição: {df_sem_mod  
elo.shape[0]}')
```

Número de registros sem aplicação contida na descrição: 792

Exclusão dos registros sem aplicação contida na descrição

Neste momento é necessário tomar uma decisão sobre o que fazer com os registros que permaneceram sem nenhuma extração na coluna **Modelo**.

Para tal decisão foi necessário observar cada um desses registros no arquivo "sem_modelo.xls" exportado e constatar que nenhum dos registros possui realmente qualquer alusão à aplicação do item descrito.

In [44]:

```
df=df[df['Modelo']!='']
```

In [45]:

```
df.reset_index(drop=True)
```

Out[45]:

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO	M
0	CHINA, REPUBLICA POP	007293# KIT TRANSMISSAO STANDARD TEMPERADO COM...	3.728000	honda cg 150 titan ks es mix fan	
1	CHINA, REPUBLICA POP	007295# KIT TRANSMISSAO STANDARD TEMPERADO COM...	3.700000	honda cg 125 titan ks es cargo	ca
2	CHINA, REPUBLICA POP	007296# KIT TRANSMISSAO STANDARD TEMPERADO COM...	3.686000	honda cg 125 fan	
3	CHINA, REPUBLICA POP	80341 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.343258	c100 biz	biz
4	CHINA, REPUBLICA POP	80364 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.826396	mirage 150	kar
5	CHINA, REPUBLICA POP	80348 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	5.232801	cbx 250 twister	21 1
6	CHINA, REPUBLICA POP	80350 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	5.969694	crf 230	2
7	CHINA, REPUBLICA POP	80373 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.245770	shineray phoenix 50cc	pl sh
8	CHINA, REPUBLICA POP	80371 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.255806	pop 110	
9	CHINA, REPUBLICA POP	80370 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.245770	pop	
10	CHINA, REPUBLICA POP	80344 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	2.950439	c125 biz	1
11	CHINA, REPUBLICA POP	80360 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.284478	hunter max 125	sui
12	CHINA, REPUBLICA POP	80342 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.255806	c100 biz	biz

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO	M
13	CHINA, REPUBLICA POP	80385 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.265841	web	sui
14	CHINA, REPUBLICA POP	80345 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	6.756764	cb 250f	25 cb
15	CHINA, REPUBLICA POP	80372 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	4.022805	riva150 dafra	
16	CHINA, REPUBLICA POP	80392 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	4.907363	xt 250 tenere	ter ye
17	CHINA, REPUBLICA POP	80397 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.136813	yes intruder 125	in :
18	CHINA, REPUBLICA POP	80395 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.136813	ybr	ye
19	CHINA, REPUBLICA POP	80346 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	4.828513	cb 300r	30 cb
20	CHINA, REPUBLICA POP	80367 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	5.034958	next 250 dafra	1 ye
21	CHINA, REPUBLICA POP	80363 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.668695	kansas 150	k
22	CHINA, REPUBLICA POP	80362 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.265841	jet 49cc	sh
23	CHINA, REPUBLICA POP	80356 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.314585	fan 125	1
24	CHINA, REPUBLICA POP	80393 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.992699	xtz 125	1
25	CHINA, REPUBLICA POP	80355 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.324620	fan 125	1

	PAIS DE ORIGEM	DESCRICAQ DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAQ	M
26	CHINA, REPUBLICA POP	80391 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	5.103773	xre 300	3
27	CHINA, REPUBLICA POP	80379 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.314585	titan	
28	CHINA, REPUBLICA POP	80378 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.314585	titan fan 150	1
29	CHINA, REPUBLICA POP	80381 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE...	3.314585	titan 160	
...	
17454	CHINA, REPUBLICA POP	10530037 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	4.865000	nxr150bros	
17455	CHINA, REPUBLICA POP	10530040 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	4.170000	crosser 150	c
17456	CHINA, REPUBLICA POP	10530041 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	4.598000	fazer 150	yε
17457	CHINA, REPUBLICA POP	10530043 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	3.751000	crypton	c yε
17458	CHINA, REPUBLICA POP	10530046 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	4.440000	fan 125	1
17459	CHINA, REPUBLICA POP	10530048 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	4.792000	broz 160	16
17460	CHINA, REPUBLICA POP	10530049 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	6.489000	ninja300 chh	kav
17461	CHINA, REPUBLICA POP	10530050 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	3.634000	pop	
17462	CHINA, REPUBLICA POP	10530051 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	4.461000	titan 160	
17463	CHINA, REPUBLICA POP	10530054 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	4.950000	fazer250 chh	yε
17464	CHINA, REPUBLICA POP	10540002 IN - KIT TRANMISSAO P/MOTOCICLETAS(C...	8.252000	nxr150bros	

	PAIS DE ORIGEM	DESCRICAQ DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAQ	M
17465	CHINA, REPUBLICA POP	10540012 IN - KIT TRANSMISSAO P/MOTOCICLETAS(C...)	8.177000	fazer250	yε
17466	CHINA, REPUBLICA POP	10540024 IN - KIT TRANSMISSAO P/MOTOCICLETAS(C...)	9.287000	xre 300	3
17467	CHINA, REPUBLICA POP	10540025 IN - KIT TRANSMISSAO P/MOTOCICLETAS(C...)	7.273000	crosser 150	c
17468	CHINA, REPUBLICA POP	10540026 IN - KIT TRANSMISSAO P/MOTOCICLETAS(C...)	7.800000	fazer 150	yε
17469	CHINA, REPUBLICA POP	10540029 IN - KIT TRANSMISSAO P/MOTOCICLETAS(C...)	8.044000	broz 160	16
17470	CHINA, REPUBLICA POP	10540031 IN - KIT TRANSMISSAO P/MOTOCICLETAS(C...)	7.463000	titan 160	
17471	CHINA, REPUBLICA POP	10540034 IN - KIT TRANSMISSAO P/MOTOCICLETAS(C...)	8.400000	fazer250 chh-uo	yε
17472	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO BIZ 100 1045 COMPOSTO D...	3.258000	biz 1045 pro honda	
17473	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO BIZ 125/POP 100 1045, C...	3.019000	biz 125 pop 1045 pro honda 14	1
17474	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO BROS 150 1045, COMPOSTO...	4.159000	broz 150 pro honda nxr corren	15
17475	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO BROS 160 1045, COMPOSTO...	4.012000	broz 160 pro honda es ks	16
17476	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO FAN 125 2009 1045, COMP...	3.607000	fan 125 pro honda cg	
17477	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO SHIN/WEB 100 36DTS 1045...	2.980000	shin web ts pro sundown	sur
17478	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO TITAN 150 1045, COMPOST...	3.648000	titan 150 pro honda cg ks es mix den	
17479	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO TITAN 160 1045, COMPOST...	3.747000	titan 160 pro honda cg	

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO	M
17480	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO TITAN 2000 1045, COMPOS...	3.620000	titan pro honda cg 125 ks es cargo	ca
17481	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO TITAN 99 1045 COMPOSTO ...	3.581000	titan 1045 pro honda cg 125	
17482	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO YBR 125 00/02 1045, COM...	3.450000	ybr 125 pro yamaha corre	ye
17483	CHINA, REPUBLICA POP	KIT DE TRANSMISSÃO AÇO YBR 125 03/05, COMPOSTO...	3.503000	ybr 125 pro yamaha factor	ye

17484 rows × 5 columns



In [46]:

```
df.shape
```

Out[46]:

(17484, 5)

Função final que transforma a DESCRIÇÃO DO PRODUTO em Modelo para classificar

In [47]:

```
def criaModelo(descricao):
    descricao=limpaDescricao(descricao)
    descricao=achaPalavraChave(descricao)
    descricao=aumentaMarca(descricao)
    return descricao
```

In [48]:

```
criaModelo(df.iloc[linha]['DESCRICAO DO PRODUTO'])
```

Out[48]:

'250 cb honda twister'

In [53]:

```
# Exporta para um arquivo
wordcloud.to_file("./imagens/wordcloud_descricoes_final.png")
```

Out[53]:

```
<wordcloud.wordcloud.WordCloud at 0x257b5059470>
```

In [54]:

```
tempotot=time.time()-initot
if tempotot>60:
    print(f'Tempo total de execução: {tempotot/60:.2f} minutos.')
else:
    print(f'Tempo total de execução: {tempotot:.2f} segundos.')
```

```
Tempo total de execução: 16.82 minutos.
```

Notebook Jupyter 4_classificarDESCRICAO

Classificação da DESCRIÇÃO obtida a partir da lista de Aplicações

Diante da necessidade de se ter uma série de registros classificados para o aprendizado supervisionado será necessária a classificação prévia de toda a base de dados para a aplicação do aprendizado de máquina que fará previsões futuras dessa classificação.

Importa as bibliotecas necessárias

In [1]:

```
import pandas as pd, numpy as np
import time
# importa o CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

In [2]:

```
# Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
```

Código executado em 20/01/2022 às 16:53

Importando a lista de Aplicações

In [3]:

```
df_aplicacoes = pd.read_csv(r'./bases/Aplicacoes.csv')
```

In [4]:

```
df_aplicacoes.head()
```

Out[4]:

APLICACOES	
0	ACELLERA ACX 250F 250
1	ACELLERA FRONTLANDER 500
2	ACELLERA FRONTLANDER 800 EFI
3	ACELLERA HOTZOO SPORT 90
4	ACELLERA QUADRILANDER 300

In [5]:

```
df_aplicacoes.shape
```

Out[5]:

```
(854, 1)
```

Observa-se dos dados acima que há 861 modelos de motocicletas disponíveis, que são as nossas possíveis aplicações dos kits de transmissão a serem importados.

Criação das Bags of Words do DataSet de aplicações

In [6]:

```
bow_aplicacoes = CountVectorizer(token_pattern='(?'u')\\b[a-zA-Z0-9\\w-]+\\b')
```

In [7]:

```
display(bow_aplicacoes)
```

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
               dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
               lowercase=True, max_df=1.0, max_features=None, min_df=1,
               ngram_range=(1, 1), preprocessor=None, stop_words=None,
               strip_accents=None, token_pattern='(?'u')\\b[a-zA-Z0-9\\w-]+\\b',
               tokenizer=None, vocabulary=None)
```

In [8]:

```
bow_aplicacoes.fit(df_aplicacoes['APLICACOES'])
```

Out[8]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
               dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
               lowercase=True, max_df=1.0, max_features=None, min_df=1,
               ngram_range=(1, 1), preprocessor=None, stop_words=None,
               strip_accents=None, token_pattern='(?u)\\b[a-zA-Z0-9\\w-]+\\b',
               tokenizer=None, vocabulary=None)
```

In [9]:

```
# Verificação das palavras
vocabulario_aplicacoes = bow_aplicacoes.vocabulary_
print(str(vocabulario_aplicacoes)[:500]+' (...) '+str(vocabulario_aplicacoes)[-500:]) # amostra
```

```
{'acellera': 85, 'acx': 87, '250f': 31, '250': 30, 'frontlander': 327, '500': 47, '800': 72, 'efi': 274, 'hotzoo': 408, 'sport': 694, '90': 78, 'quadrilander': 608, '300': 35, '400': 40, '600': 54, 'sportlander': 696, '150r': 20, '150': 19, '250xr': 32, '350zx': 38, '350': 37, '450tr': 44, '450': 42, 'adly': 89, 'atv': 122, 'jaguar': 427, 'rt': 654, 'agrale': 96, 'city': 196, 'dakar': 222, 'elefant': 276, 're': 633, 'explorer': 296, 'force': 321, 'junior': 443, 'sst': 707, 'super': 723, 'sxt': 7 (...), 'tzr': 784, 'v-max': 787, '1680': 22, 'v-star': 790, 'virago': 807, '535': 51, '450f': 43, 'xf50x': 850, 'xj6': 851, 'xjr': 852, 'xr180': 862, 'xs1100': 867, 'xs': 866, 'xs400': 868, 'xs500': 869, 'xs650': 870, 'xs750': 871, 'xs850': 872, 'xt': 873, '225': 26, '660r': 59, '660': 58, 'tenere': 738, '1200z': 12, '660z': 60, 'crossover': 210, 'xz': 877, 'yfm': 883, '700r': 64, 'yfs': 884, 'yz': 887, '85lw': 77, 'yzf': 888, '600r': 55, 'r1': 613, 'r6': 620, 'yzr': 889, 'zongshen': 900, 'zs': 903}
```

In [10]:

```
# Verificação do número de termos no vocabulário de aplicações
len(bow_aplicacoes.vocabulary_)
```

Out[10]:

905

In [11]:

```
# Transformação em vetores binários
X_bow_aplicacoes = bow_aplicacoes.fit_transform(df_aplicacoes[ 'APLICACOES' ])
print(f'{X_bow_aplicacoes[:2]}\n        (...)\\n{X_bow_aplicacoes[-2:]}' ) # a
mostra
```

```
(0, 30)      1
(0, 31)      1
(0, 87)      1
(0, 85)      1
(1, 47)      1
(1, 327)     1
(1, 85)      1
        ...
(0, 903)     1
(0, 900)     1
(0, 13)      1
(1, 903)     1
(1, 900)     1
(1, 25)      1
```

In [12]:

```
type(X_bow_aplicacoes)
```

Out[12]:

```
scipy.sparse.csr.csr_matrix
```

In [13]:

```
matrix_aplicacoes = bow_aplicacoes.transform(df_aplicacoes[ 'APLICACOES' ])
```

In [14]:

```
print(f'{matrix_aplicacoes[:2]}\n        (...)\\n{matrix_aplicacoes[-2:]}' ) # a
mostra
```

```
(0, 30)      1
(0, 31)      1
(0, 85)      1
(0, 87)      1
(1, 47)      1
(1, 85)      1
(1, 327)     1
        ...
(0, 13)      1
(0, 900)     1
(0, 903)     1
(1, 25)      1
(1, 900)     1
(1, 903)     1
```

In [15]:

```
print(matrix_aplicacoes.shape)
```

(854, 905)

In [16]:

```
print(matrix_aplicacoes.toarray())
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 1 0]]
```

In [17]:

```
df1=pd.DataFrame(matrix_aplicacoes.toarray())
df1.head()
```

Out[17]:

0	1	2	3	4	5	6	7	8	9	...	895	896	897	898	899	900	901	902
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

5 rows × 905 columns

In [18]:

```
# Totalização do número de termos de cada vocáculo
df_sum=df1.sum(axis=0)
df_sum.head()
```

Out[18]:

```
0      9
1      1
2      1
3      1
4      1
dtype: int64
```

In [19]:

```
# Sumário do dataset
df_sum.describe()
```

Out[19]:

```
count    905.000000
mean      2.501657
std       5.666452
min      1.000000
25%     1.000000
50%     1.000000
75%     2.000000
max     97.000000
dtype: float64
```

In [20]:

```
# Identificação dos valores máximos
print(df_sum.sort_values().head(), df_sum.sort_values().tail())
```

```
452    1
551    1
552    1
554    1
556    1
dtype: int64 30      36
459    37
599    40
406    79
878    97
dtype: int64
```

In [21]:

```
# Identificação dos valores com uma ocorrência
df_sum[df_sum==1].sample(5)
```

Out[21]:

```
289    1
251    1
479    1
894    1
387    1
dtype: int64
```

Diante da necessidade de se ter uma série de registros classificados para o aprendizado supervisionado e após a determinação dos 858 modelos de aplicações disponíveis, vamos classificar baseados nas seguintes observações realizadas na análise exploratória dos dados:

1. Os modelos de motocicletas têm nomes que os distinguem em grande parte dos casos;
2. Dos 913 termos, 655 aparecem apenas uma vez, o que implica que sua ocorrência já deve classificar o item;
3. O termo com maior número de ocorrências aparece 97 vezes (YAMAHA), sendo o menos distintivo de todos.

In [22]:

```
# Função para pegar a chave pelo valor, dado que valor é único.
def pegaChave(v):
    for chave, valor in vocabulario_aplicacoes.items():
        if v == valor:
            return chave
    return "Não existe chave para esse valor."
```

In [23]:

```
print(pegaChave(881).upper())
```

YBR125

In [24]:

```
# Criação do dicionário chaves invertendo chave e valor do vocabulário
chaves = dict((v,k) for k,v in vocabulario_aplicacoes.items())
```

In [25]:

```
# Termo do vocabulário pelo índice
chaves[887].upper()
```

Out[25]:

'YZ'

In [26]:

```
# Índice pelo termo do vocabulário
vocabulario_aplicacoes['yamaha']
```

Out[26]:

878

In [27]:

```
# Número de ocorrências pelo termo do vocabulário
df_sum[vocabulario_aplicacoes['yamaha']]
```

Out[27]:

97

In [28]:

```
# Função para determinar a aplicação caso a contagem do termo seja 1
def achaAplicacao1(modelo):
    # pega os termos do modelo e transforma em uma lista
    modelolst = modelo.lower().split()
    # para cada termo
    for mod in modelolst:
        try:
            # l é o índice termo no vocabulário de aplicações
            l = vocabulario_aplicacoes[mod]
            # se a soma de todos os termos do índice i do vocabulário for 1
            if df_sum[l]==1:
                # pegar o índice dessa linha
                # que será onde o valor 1 aparece na coluna chamada Linha d
                o df1
                i = int(df1[df1[l]==1].index.values)
                # retornar a descrição da aplicação dessa linha
                return df_aplicacoes['APLICACOES'][i]
        except:
            continue
    return 'XXX'
```

In [29]:

```
print(achaAplicacao1('MOTO TEMOS OUTRO ZING'))
```

KIMCO ZING

Importa os dados já tratados

In [30]:

```
# Importa base de dados com os modelos já determinados para um dataframe
df = pd.read_excel('./bases/dataframe_modelos.xlsx')
```

In [31]:

```
df.sample(4)
```

Out[31]:

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO
11027	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO , MARCA RIFFEL, TITANIUM (1...	3.750	biz 125
4255	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO , MARCA RIFFEL, TITANIUM (1...	4.060	ybr 125 factor
5884	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO , MARCA RIFFEL, TOP (1045) ...	14.950	cb 300r
7989	CHINA, REPUBLICA POP	10530035 IN KIT TRANSMISSAO P/MOTOCICLETAS(COR...	6.459	cb 300



In [32]:

```
df.iloc[:, -2:].head()
```

Out[32]:

	DESCRICAO	Modelo
0	honda cg 150 titan ks es mix fan	150 cg fan honda titan
1	honda cg 125 titan ks es cargo	125 cargo cg honda titan
2	honda cg 125 fan	125 cg fan honda
3	c100 biz	biz c100 honda
4	mirage 150	150 kasinski mirage

In [33]:

```
# Verifica o tamnanho do dataframe
df.shape
```

Out[33]:

```
(17484, 5)
```

Classificando segundo a lista de Aplicações

In [34]:

```
df['APLICACAO']=df['Modelo'].apply(achaAplicacao1)
```

In [35]:

```
df.iloc[:, -3: ].head(10)
```

Out[35]:

	DESCRICAO	Modelo	APLICACAO
0	honda cg 150 titan ks es mix fan	150 cg fan honda titan	HONDA CG FAN
1	honda cg 125 titan ks es cargo	125 cargo cg honda titan	HONDA CG TIT TITAN 125 150 160
2	honda cg 125 fan	125 cg fan honda	HONDA CG FAN
3	c100 biz	biz c100 honda	HONDA BIZ C100 125 C125
4	mirage 150	150 kasinski mirage	XXX
5	cbx 250 twister	250 cbx honda twister	HONDA TWISTER CBX 250
6	crf 230	230 crf honda	XXX
7	shineray phoenix 50cc	50 phoenix shineray	XXX
8	pop 110	honda pop	HONDA POP
9	pop	honda pop	HONDA POP

In [36]:

```
df[df['APLICACAO']=='XXX'].iloc[:, -3: ].head(10)
```

Out[36]:

	DESCRICAO	Modelo	APLICACAO
4	mirage 150	150 kasinski mirage	XXX
6	crf 230	230 crf honda	XXX
7	shineray phoenix 50cc	50 phoenix shineray	XXX
11	hunter max 125	125 hunter max sundown	XXX
14	cb 250f	250 250f cb honda	XXX
16	xt 250 tenere	250 tenere xt yamaha	XXX
22	jet 49cc	jet shineray	XXX
24	xtz 125	125 xtz	XXX
30	fazer 150	150 fazer yamaha	XXX
38	bros150	150	XXX

In [37]:

```
print('Registros sem classificação: ' + str(df[df['APLICACAO']=='XXX'].iloc
c[:, -3:].shape[0]))
print('Registros com classificação: ' + str(df[df['APLICACAO']!='XXX'].iloc
[:, -3:].shape[0]))
print('Total de Registros: ' + str(df.shape[0]))
```

Registros sem classificação: 2867
 Registros com classificação: 14617
 Total de Registros: 17484

Observa-se que após a classificação pelos termos únicos restaram cerca de 3.000 linhas, havendo mais de 15.000 registros já classificados.

Classifica o modelo como aplicação se os termos forem exatamente iguais

In [38]:

```
def achaAplicacao2(modelo):
    modelolst=modelo.lower().split() # cria lista com termos do modelo
    aplicacoes_temp=[] # inicializa a lista de saída
    for aplicacao in df_aplicacoes['APLICACOES']: # para cada aplicação
        aplicacaolst = aplicacao.lower().split() # cria lista
        if all(termos in aplicacaolst for termos in modelolst): # se a aplicação contém o modelo
            aplicacoes_temp.append(aplicacao) # adiciona na lista de saída
    if len(aplicacoes_temp)==0:
        return 'XXX' # retorna a saída
    elif len(aplicacoes_temp)==1:
        return aplicacoes_temp[0] # retorna a saída
    else:
        return aplicacoes_temp # retorna a saída
```

In [39]:

```
# df temporário filtrado pelos não classificados
dftemp=df.iloc[:, -2:][df['APLICACAO']=='XXX']
dftemp.shape
```

Out[39]:

(2867, 2)

In [40]:

```
dftemp.head()
```

Out[40]:

	Modelo	APLICACAO
4	150 kasinski mirage	XXX
6	230 crf honda	XXX
7	50 phoenix shineray	XXX
11	125 hunter max sundown	XXX
14	250 250f cb honda	XXX

In [41]:

```
# aplica a função achaAchaAplicacao2
dftemp=dftemp.assign(APLICACAO=dftemp['Modelo'].apply(achaAplicacao2))
```

In [42]:

```
dftemp.sample(5)
```

Out[42]:

	Modelo	APLICACAO
5011	230 crf honda	HONDA CRF 230 230F 250 250F
8463	150 fazer yamaha	YAMAHA FAZER YS150 150
12781	150 fazer yamaha	YAMAHA FAZER YS150 150
12872	300 cb honda rx	XXX
5056	125 xtz	YAMAHA XTZ 125

In [43]:

```
# quantidade de registros sem classificação
dftemp[dftemp['APLICACAO']=='XXX'].shape
```

Out[43]:

(297, 2)

In [44]:

```
# atualiza o dataframe df com as alterações feitas em dftemp
df.update(dftemp)
```

In [45]:

```
df.iloc[:, -3:].sample(5)
```

Out[45]:

	DESCRICAO	Modelo	APLICACAO
16336	xtz 125	125 xtz	YAMAHA XTZ 125
16309	cg 125 cargo fan	125 cargo cg fan honda	HONDA CG FAN
9294	cg 150 titan fan start cargo	150 cargo cg fan honda titan	HONDA CG FAN
15136	nxr 160 broz xre	160 broz honda nxr xr xre	HONDA NXR 150 160 BROZ
10206	xlr 125	125 xlr	HONDA XLR

In [46]:

```
df[df['APLICACAO']=='XXX'].iloc[:, -3:].sample(5)
```

Out[46]:

	DESCRICAO	Modelo	APLICACAO
15998	cbx 200 xr	200 cbx honda xr	XXX
13267	max 125 hunter p21b	125 hunter max sundown	XXX
2740	hunter max125	125 hunter max sundown	XXX
12178	p21b max 125 hunter	125 hunter max sundown	XXX
13867	max 125 hunter mot	125 hunter max sundown	XXX

In [47]:

```
print('Registros sem classificação: ' + str(df[df['APLICACAO']=='XXX'].iloc[:, -3:].shape[0]))
print('Registros com classificação: ' + str(df[df['APLICACAO']!='XXX'].iloc[:, -3:].shape[0]))
print('Total de Registros: ' + str(df.shape[0]))
```

```
Registros sem classificação: 297
Registros com classificação: 17187
Total de Registros: 17484
```

Observa-se que após a classificação pelos termos únicos restaram cerca de 600 linhas. Para fins de utilização no aprendizado de classificação, os quase 17.000 registros restantes são suficientes para comparação com os outros modelos e determinar a melhor classificação.

Exportando o DataFrame

Exportando para um arquivo CSV

In [48]:

```
df.to_csv(r'./bases/dataframe_modelos_class0.csv', index = False, header = True)
```

Exportando para um arquivo de planilha do Excel

In [49]:

```
df.to_excel(r'./bases/dataframe_modelos_class0.xlsx', index = False, header = True)
```

In [50]:

```
tempotot=time.time()-initot
if tempotot>60:
    print(f'Tempo total de execução: {tempotot/60:.2f} minutos.')
else:
    print(f'Tempo total de execução: {tempotot:.2f} segundos.')
```

Tempo total de execução: 12.02 segundos.

Notebook Jupyter 5_NLP_modeloClassificador

Classificação da Aplicação por aprendizado de máquina

Importando bibliotecas

In [1]:

```
import pandas as pd, numpy as np, time
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
```

In [2]:

```
# Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
```

Código executado em 20/01/2022 às 16:54

Importando a lista de Aplicações

In [3]:

```
df_aplicacoes = pd.read_csv('r'./bases/Aplicacoes.csv')
```

In [4]:

```
df_aplicacoes.head(2)
```

Out[4]:

APLICACOES	
0	ACELLERA ACX 250F 250
1	ACELLERA FRONTLANDER 500

In [5]:

```
df_aplicacoes.tail(2)
```

Out[5]:

APLICACOES	
852	ZONGSHEN ZS 125
853	ZONGSHEN ZS 200

In [6]:

```
df_aplicacoes.shape
```

Out[6]:

```
(854, 1)
```

CountVectorizer

CountVectorizer do DataSet das Aplicações

In [7]:

```
# Criação da função CountVectorizer
cvta = CountVectorizer(strip_accents='ascii', lowercase=True)
X_cvta = cvta.fit_transform(df_aplicacoes['APLICACOES'])
```

Treinando os Modelos com o DataSet Aplicações

Definindo os parâmetros

Utilizaremos toda a base no treinamento, pois a intenção é criar uma função de classificação para um dataset onde a classificação é inexistente.

In [8]:

```
X_train=X_cvta.toarray()
y_train=np.array(df_aplicacoes['APLICACOES'])
y_train1=df_aplicacoes['APLICACOES'].index.to_numpy()
```

Modelo LinearSVC

In [9]:

```
# Criando modelo
clfsvc = LinearSVC()
# Treinamento do modelo
clfsvc.fit(X_train, y_train)
```

Out[9]:

```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
           intercept_scaling=1, loss='squared_hinge', max_iter=1000,
           multi_class='ovr', penalty='l2', random_state=None, tol=
0.0001,
           verbose=0)
```

Função de classificação LinearSVC

A função para utilização do modelo, recebe a descrição filtrada Modelo e retorna a aplicação.

In [10]:

```
def classificaAplicacaoSVC(modelo):
    novo_cvta = cvta.transform(pd.Series(modelo))
    aplicacao = clfsvc.predict(novo_cvta)[0]
    return aplicacao
```

No final o nosso resultado mostrando (Modelo: descrição filtrada para o modelo e Aplicação: aplicação prevista).

In [11]:

```
# Lista de exemplos de novos produtos
modelos = ['150 CG HONDA TITAN',
            '125 CARGO CG HONDA TITAN',
            'BIZ C100 HONDA',
            '100 HONDA BIZ',
            '100 BIZ BRAVO HONDA',
            '125 YBR GT YAMAHA',
            '250F TWISTER HONDA']
# Loop for para fazer a predição do departamento de novos produtos
for modelo in modelos:
    print('Modelo:', modelo, ' Aplicação:', classificaAplicacaoSVC(modelo))
```

Modelo: 150 CG HONDA TITAN Aplicação: HONDA CG TIT TITAN 125
150 160
Modelo: 125 CARGO CG HONDA TITAN Aplicação: HONDA CG TIT TITA
N 125 150 160
Modelo: BIZ C100 HONDA Aplicação: HONDA BIZ C100 125 C125
Modelo: 100 HONDA BIZ Aplicação: HONDA BIZ C100 125 C125
Modelo: 100 BIZ BRAVO HONDA Aplicação: SHINERAY BRAVO 200
Modelo: 125 YBR GT YAMAHA Aplicação: SUZUKI GT
Modelo: 250F TWISTER HONDA Aplicação: HONDA TWISTER CBX 250

Modelo Multinomial Naive Bayes

In [12]:

```
# Criando modelo
clfMNB = MultinomialNB()
# Treinamento do modelo
clfMNB.fit(X_train, y_train1)
```

Out[12]:

MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

Função de classificação

A função para utilização do modelo, recebe a descrição filtrada Modelo e retorna a aplicação.

In [13]:

```
def classificaAplicacaoMNB(modelo):
    novo_cvta = cvta.transform(pd.Series(modelo))
    aplicacao = df_aplicacoes['APLICACOES'][clfMNB.predict(novo_cvta)[0]]
    return aplicacao
```

No final o nosso resultado mostrando (Modelo: descrição filtrada para o modelo e Aplicação: aplicação prevista).

In [14]:

```
# Lista de exemplos de novos produtos
modelos = ['150 CG FAN HONDA TITAN',
            '125 CARGO CG HONDA TITAN',
            'BIZ C100 HONDA',
            '100 HONDA BIZ',
            '100 BIZ BRAVO HONDA',
            '125 YBR GT YAMAHA',
            '250F TWISTER HONDA']
# Loop for para fazer a predição do departamento de novos produtos
for modelo in modelos:
    print('Modelo:', modelo, 'Aplicação:', classificaAplicacaoMNB(modelo))
```

Modelo: 150 CG FAN HONDA TITAN Aplicação: HONDA CG TIT TITAN 1
25 150 160
Modelo: 125 CARGO CG HONDA TITAN Aplicação: HONDA CG TIT TITAN
125 150 160
Modelo: BIZ C100 HONDA Aplicação: HONDA BIZ C100 125 C125
Modelo: 100 HONDA BIZ Aplicação: HONDA BIZ C100 125 C125
Modelo: 100 BIZ BRAVO HONDA Aplicação: HONDA BIZ C100 125 C125
Modelo: 125 YBR GT YAMAHA Aplicação: YAMAHA FACTOR YBR 125 YBR
125
Modelo: 250F TWISTER HONDA Aplicação: HONDA CB 250 250F

Modelo de Regressão Logística

In [15]:

```
# Criando modelo
clfLgr = LogisticRegression(solver='lbfgs', multi_class='multinomial')
# Treinamento do modelo
clfLgr.fit(X_train, y_train)
```

Out[15]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='multinomial',
                   n_jobs=None, penalty='l2', random_state=None, solver='lbfgs',
                   tol=0.0001, verbose=0, warm_start=False)
```

Função de classificação

A função para utilização do modelo, recebe a descrição filtrada Modelo e retorna a aplicação.

In [16]:

```
def classificaAplicacaoLGR(modelo):
    novo_cvta = cvta.transform(pd.Series(modelo))
    aplicacao = clflgr.predict(novo_cvta)[0]
    return aplicacao
```

No final o nosso resultado mostrando (Modelo: descrição filtrada para o modelo e Aplicação: aplicação prevista).

In [17]:

```
# Lista de exemplos de novos produtos
modelos = ['150 CG FAN HONDA TITAN',
           '125 CARGO CG HONDA TITAN',
           'BIZ C100 HONDA',
           '100 HONDA BIZ',
           '100 BIZ BRAVO HONDA',
           '125 YBR GT YAMAHA',
           '250F TWISTER HONDA']
# Loop for para fazer a predição do departamento de novos produtos
for modelo in modelos:
    print('Modelo:', modelo, 'Aplicação:', classificaAplicacaoLGR(modelo))
```

Modelo: 150 CG FAN HONDA TITAN Aplicação: HONDA CG TIT TITAN 1
25 150 160

Modelo: 125 CARGO CG HONDA TITAN Aplicação: HONDA CG TIT TITAN
125 150 160

Modelo: BIZ C100 HONDA Aplicação: HONDA BIZ C100 125 C125

Modelo: 100 HONDA BIZ Aplicação: HONDA BIZ C100 125 C125

Modelo: 100 BIZ BRAVO HONDA Aplicação: HONDA BIZ C100 125 C125

Modelo: 125 YBR GT YAMAHA Aplicação: YAMAHA FACTOR YBR 125 YBR
125

Modelo: 250F TWISTER HONDA Aplicação: HONDA TWISTER CBX 250

Utilização do modelo

Carregando dataset

In [18]:

```
# Importa base de dados com os modelos já determinados para um dataframe
df = pd.read_excel(r'./bases/dataframe_modelos_class0.xlsx')
df.iloc[:, -2:].head()
```

Out[18]:

	Modelo	APLICACAO
0	150 cg fan honda titan	HONDA CG FAN
1	125 cargo cg honda titan	HONDA CG TIT TITAN 125 150 160
2	125 cg fan honda	HONDA CG FAN
3	biz c100 honda	HONDA BIZ C100 125 C125
4	150 kasinski mirage	KASINSKI MIRAGE 150 250

In [19]:

```
# Verifica o tamnanho do dataframe
df.shape
```

Out[19]:

(17484, 6)

Classificando os modelos do DataSet

In [20]:

```
# Modelo Linear SVC
ini=time.time()
now = time.strftime("%H:%M:%S", time.localtime(time.time()))
t1=5/df.shape[0]*1000 # tempo para cada mil registros
print(f"Hora de inicio: {now}")
print(f"Tempo estimado: {int(t1*df.shape[0]/1000)} segundos")

df['APLICACAOSVC']=df['Modelo'].apply(classificaAplicacaoSVC)

now = time.strftime("%H:%M:%S", time.localtime(time.time()))
fim=time.time()
print("Hora de término: " + str(now))
print("Tempo decorrido: " + str(round((fim-ini),1)) + " segundos.")
```

Hora de inicio: 16:54:08
 Tempo estimado: 5 segundos
 Hora de término: 16:54:13
 Tempo decorrido: 4.7 segundos.

In [21]:

```
# Modelo Multinomial NB
ini=time.time()
now = time.strftime("%H:%M:%S", time.localtime(time.time()))
t1=53/df.shape[0]*1000 # tempo para cada mil registros
print(f"Hora de início: {now}")
print(f"Tempo estimado: {int(t1*df.shape[0]/1000)} segundos")

df['APLICACAOMNB']=df['Modelo'].apply(classificaAplicacaoMNB)

now = time.strftime("%H:%M:%S", time.localtime(time.time()))
fim=time.time()
print("Hora de término:" + str(now))
print("Tempo decorrido: " + str(round((fim-ini),1)) + " segundos.")
```

Hora de início: 16:54:13
 Tempo estimado: 53 segundos
 Hora de término: 16:55:06
 Tempo decorrido: 52.9 segundos.

In [22]:

```
# Modelo Regressão Logística
ini=time.time()
now = time.strftime("%H:%M:%S", time.localtime(time.time()))
t1=68/df.shape[0]*1000 # tempo para cada mil registros
print(f"Hora de início: {now}")
print(f"Tempo estimado: {int(t1*df.shape[0]/1000)} segundos")

df['APLICACAOLGR']=df['Modelo'].apply(classificaAplicacaoLGR)

now = time.strftime("%H:%M:%S", time.localtime(time.time()))
fim=time.time()
print("Hora de término:" + str(now))
print("Tempo decorrido: " + str(round((fim-ini),1)) + " segundos.")
```

Hora de início: 16:55:06
 Tempo estimado: 68 segundos
 Hora de término: 16:56:06
 Tempo decorrido: 60.2 segundos.

In [23]:

```
df.shape
```

Out[23]:

(17484, 9)

In [24]:

```
df.iloc[:, -6:].sample(5)
```

Out[24]:

	DESCRICAO	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB
4530	cg 125 cargo fan titan	125 cargo cg fan honda titán	HONDA CG FAN	HONDA CG FAN	HONDA CG TIT TITAN 125 150 160
13880	520vo ht crf 230	230 crf honda	HONDA CRF 230 230F 250 250F	HONDA CRF 230 230F 250 250F	HONDA CRF 230 230F 250 250F
17237	todos acondicionado única titan 125 fan	125 fan honda titán	HONDA CG FAN	HONDA CG FAN	HONDA CG TIT TITAN 125 150 160
6817	tkbrybrb factor ybr 125	125 factor yamaha ybr	YAMAHA FACTOR YBR 125 YBR125	YAMAHA FACTOR YBR 125 YBR125	YAMAHA FACTOR YBR 125 YBR125
14379	falcon 400	400 falcon	HONDA NX 400 FALCON	HONDA NX 400 FALCON	HONDA NX 400 FALCON

◀ ▶

Comparação das classificações

In [25]:

```
# Definição dos filtros
f1 = df['APLICACAOSVC'] == df['APLICACAOMNB'] # Modelos SVC e MNB iguais
f2 = df['APLICACAOSVC'] == df['APLICACAOLGR'] # Modelos SVC e LGR iguais
f3 = df['APLICACAOSVC'] == df['APLICACAO'] # Modelo e SVC iguais
f4 = df['APLICACAOMNB'] == df['APLICACAOLGR'] # Modelos MNB e LGR iguais
f5 = df['APLICACAOMNB'] == df['APLICACAO'] # Modelo e MNB iguais
f6 = df['APLICACAOLGR'] == df['APLICACAO'] # Modelo e LGR iguais
```

In [26]:

```
# Quantidade de registros divergentes entre os modelos SVC e Multinomial NB
df[~f1].shape
```

Out[26]:

(3794, 9)

In [27]:

```
# Quantidade de registros divergentes entre os modelos SVC e Regressão Logística
df[~f2].shape
```

Out[27]:

(3230, 9)

In [28]:

```
# Quantidade de registros divergentes entre a extração e o modelo SVC
df[~f3].shape
```

Out[28]:

(1620, 9)

In [29]:

```
# Quantidade de registros divergentes entre os modelos MNB e Regressão Logística
df[~f4].shape
```

Out[29]:

(1166, 9)

In [30]:

```
# Quantidade de registros divergentes entre a extração e o Modelo MNB
df[~f5].shape
```

Out[30]:

(3622, 9)

In [31]:

```
# Quantidade de registros divergentes entre a extração e o Modelo LGR
df[~f6].shape
```

Out[31]:

(3014, 9)

In [32]:

```
# Quantidade de registros divergente entre os quatro
df[~(f1 & f2 & f3)].shape
```

Out[32]:

(4585, 9)

In [33]:

```
# Quantidade de registros iguais nos quatro
df[f1 & f2 & f3].shape
```

Out[33]:

(12899, 9)

Em virtude do tempo de processamento maior e também da maior quantidade de divergências, decidiu-se por abandonar o uso da modelo de regressão logística.

In [34]:

```
df.drop('APLICACAOLGR', axis=1, inplace=True)
```

In [35]:

```
df[df['APLICACAOSVC']!=df['APLICACAOMNB']].iloc[:5,-5:]
```

Out[35]:

	DESCRICAO	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB
0	honda cg 150 titan ks es mix fan	150 cg fan honda titan	HONDA CG FAN	HONDA CG FAN	HONDA CG TIT TITAN 125 150 160
2	honda cg 125 fan	125 cg fan honda	HONDA CG FAN	HONDA CG FAN	HONDA CG 125
11	hunter max 125	125 hunter max sundown		XXX SUNDOWN MAX	SUNDOWN HUNTER
17	yes intruder 125	125 intruder suzuki yes	SUZUKI YES EN 125	SUZUKI INTRUDER	SUZUKI YES EN 125
22	jet 49cc	jet shineray	['SHINERAY JET 125', 'SHINERAY JET 50']	SHINERAY JET 50	SHINERAY JET 125

In [36]:

```
df['DESCRICAO DO PRODUTO'].iloc[11]
```

Out[36]:

'80360 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE, COROA E PINHÃ
O PARA MOTOCICLETA HUNTER / MAX 125, MARCA ALLEN.'

Melhorando a classificação

Para chegar à classificação final a ser utilizada na função de classificação, utilizaremos algumas regras:

1. SE APPLICACAOMNB==APPLICACAOSVC ==> APPLICACAOFIM=APPLICACAOSVC
2. SE APPLICACAOMNB!=APPLICACAOSVC
 - A. SE APPLICACAO=='XXX' ==> APPLICACAOFIM=[APPLICACAOSVC,APPLICACAOMNB]
 - B. SE APPLICACAO==APPLICACAOSVC ==> APPLICACAOFIM=APPLICACAOSVC
 - C. SE APPLICACAO==APPLICACAOMNB ==> APPLICACAOFIM=APPLICACAOMNB
 - D. SE APPLICACAO for uma lista:
 - SE APPLICACAOSVC estiver na lista ==> APPLICACAOFIM=APPLICACAOSVC
 - SE APPLICACAOMNB estiver na lista ==> APPLICACAOFIM=APPLICACAOSMNB
 - SENÃO ==> APPLICACAOFIM=LISTA+[APPLICACAOSVC,APPLICACAOMNB]
 - E. SE APPLICACAO!=APPLICACAOSVC!=APPLICACAOMNB ==> APPLICACAOFIM=[APPLICACAO,APPLICACAOSVC,APPLICACAOMNB]

In [37]:

```
df=df.assign(APPLICACAOFIM=df.APPLICACAOMNB.tolist())
df.shape
```

Out[37]:

(17484, 9)

In [38]:

```
df.iloc[:, -5:].sample(10)
```

Out[38]:

	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB	APLICACAOPRC
15752	300 xre	HONDA XRE 300	HONDA XRE 300	HONDA XRE 300	HOND/ 300
4949	250 ['ACELLERA ACX 250F 250', 'ACELLERA SPORTLANDE...']		KTM SX 250	AMAZONAS AME-250	AMAZ AV
9789	150 cargo cg fan honda titán	HONDA CG FAN	HONDA CG FAN	HONDA CG TIT TITAN 125 150 160	HONDA C TITAN 125
13924	crypton yamaha	YAMAHA CRYPTON	YAMAHA CRYPTON	YAMAHA CRYPTON	YA CRY
17426	125 yamaha ybr ybr125	YAMAHA FACTOR YBR 125 YBR125	YAMAHA FACTOR YBR 125 YBR125	YAMAHA FACTOR YBR 125 YBR125	YA FACTOR 125 YB
11212	250 lander xtz yamaha	YAMAHA LANDER XTZ 250	YAMAHA LANDER XTZ 250	YAMAHA LANDER XTZ 250	YA LANDER
12447	150 broz honda	HONDA NXR 150 160 BROZ	HONDA NXR 150 160 BROZ	HONDA NXR 150 160 BROZ	HOND/ 150 160
12919	crypton yamaha	YAMAHA CRYPTON	YAMAHA CRYPTON	YAMAHA CRYPTON	YA CRY
6645	125 factor yamaha ybr	YAMAHA FACTOR YBR 125 YBR125	YAMAHA FACTOR YBR 125 YBR125	YAMAHA FACTOR YBR 125 YBR125	YA FACTOR 125 YB
7302	250 fazer yamaha	YAMAHA FAZER YS250 250	YAMAHA FAZER YS250 250	YAMAHA FAZER YS250 250	YAMAHA F YS250



In [39]:

```

colindex = df.columns.get_loc("APLICACAOFIM")
for i in range(df.shape[0]):
    #print(i)
    # se APLICACAOSVC==APLICACAOMNB ==> já aplicado pelo df.assign
    # se APLICACAOSVC!=APLICACAOMNB
    if df['APLICACAOSVC'][i]!=df['APLICACAOMNB'][i]:
        # se APLICAÇÃO for sem valor ('XXX') ==> APLICACAOFIM=[APLICACAOSVC,APLICACAOMNB]
        if df['APLICACAO'][i]=='XXX':
            # APLICACAOFIM será a lista com os dois valores
            df.iloc[i,colindex]=str([df['APLICACAOSVC'][i], df['APLICACAOMNB'][i]])
            continue # passa para o próximo i
        elif df['APLICACAO'][i]==df['APLICACAOSVC'][i]:
            df.iloc[i,colindex]=df['APLICACAOSVC'][i]
            continue # passa para o próximo i
        elif df['APLICACAO'][i]==df['APLICACAOMNB'][i]:
            df.iloc[i,colindex]=df['APLICACAOMNB'][i]
            continue # passa para o próximo i
        elif "," in df['APLICACAO'][i]: # se tiver vírgula, a APLICACAO é uma lista
            aplictemp=df['APLICACAO'][i].replace("[", "").replace(", ", ";").replace(", ;").replace("'", "").replace("[", "")
            aplictemp=aplictemp.split(';')
            # se APLICACAOSVC estiver na lista
            if df['APLICACAOSVC'][i] in aplictemp:
                df.iloc[i,colindex]=df['APLICACAOSVC'][i]
                continue # passa para o próximo i
            # se APLICACAOMNB estiver na lista
            elif df['APLICACAOMNB'][i] in aplictemp:
                df.iloc[i,colindex]=df['APLICACAOMNB'][i]
                continue # passa para o próximo i
            else:
                aplictemp.append(df['APLICACAOSVC'][i])
                aplictemp.append(df['APLICACAOMNB'][i])
                df.iloc[i,colindex]=str(aplictemp)
                continue # passa para o próximo i
        else: # caso seja um valor (diferente de ambos os classificadores)
            # APLICACAO!=APLICACAOSVC!=APLICACAOMNB ==> APLICACAOFIM=[APLICACAO,APLICACAOSVC,APLICACAOMNB]
            df.iloc[i,colindex]=str([df['APLICACAO'][i],df['APLICACAOSVC'][i], df['APLICACAOMNB'][i]])
            continue # passa para o próximo i

```

Recomparação das classificações

In [40]:

```

# Definição dos filtros
f1 = df['APLICACAOSVC']==df['APLICACAOMNB'] # Modelos SVC e MNB iguais
f2 = df['APLICACAOSVC']==df['APLICACAOFIM'] # Modelo manual e SVC iguais
f3 = df['APLICACAOMNB']==df['APLICACAOFIM'] # Modelo manual e MNB iguais

```

In [41]:

```
# Quantidade de registros divergentes entre os modelos SVC e Multinomial NB
df[~f1].shape
```

Out[41]:

(3794, 9)

In [42]:

```
# Quantidade de registros divergentes entre a extração e o modelo SVC
df[~f2].shape
```

Out[42]:

(907, 9)

In [43]:

```
# Quantidade de registros divergentes entre a extração e o modelo SVC
df[~f3].shape
```

Out[43]:

(3049, 9)

In [44]:

```
# Quantidade de registros divergente entre os três
df[~(f1 & f2)].shape
```

Out[44]:

(3794, 9)

In [45]:

```
# Quantidade de registros iguais nos três
df[f1 & f2].shape
```

Out[45]:

(13690, 9)

In [46]:

```
# Registros ainda não definidos (Lista de opções)
# O filtro definirá True se APLICACAOFIM iniciar com '[' ou False caso contrário.
filtro=[]
for i, aplicacao in enumerate(df['APLICACAOFIM']):
    if aplicacao[0]=='[':
        filtro.append(True)
    else:
        filtro.append(False)
```

In [47]:

```
df[filtro].iloc[:, -5: ].head()
```

Out[47]:

	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB	APLICACAOFII
11	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER	['SUNDOWI MAX 'SUNDOWI HUNTER
103	200 cbx honda xr	XXX	HONDA XR	HONDA STRADA CBX 200	['HONDA XR 'HOND. STRADA CB. 200
124	150 fazer sm yamaha	XXX	HUSQVARNA SM	YAMAHA FAZER YS150 150	['HUSQVARN. SM', 'YAMAH. FAZER YS15 150
282	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER	['SUNDOWI MAX 'SUNDOWI HUNTER
304	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER	['SUNDOWI MAX 'SUNDOWI HUNTER

In [48]:

```
df[filtro].shape
```

Out[48]:

(162, 9)

Classificando pela quantidade de termos em comum

In [49]:

```
# determina o(s) índice(s) do valor máximo de uma Lista
def indicesMaxLista(lista):
    maximo=max(lista)
    indices=[]
    for i, elemento in enumerate(lista):
        if elemento==maximo:
            indices.append(i)
    return indices
```

In [50]:

```
colindex = df.columns.get_loc("APLICACAOFIM") # índice da coluna
for i, row in df[filtro].iterrows(): #para cada linha do dataset
    qtds=[]
    aplicMOD=row[ 'Modelo'].upper().split()
    aplicFIM=row[ 'APLICACAOFIM'].replace("[","").replace(", ", ";").replace(
",", ";").replace("", "").replace("]", "").split(';')
    for a, aplicacao in enumerate(aplicFIM):
        aplicFIMlista=aplicacao.split()
        qtds.append(len(set(aplicFIMlista).intersection(set(aplicMOD)))) #determinar len da intersecção
    indicesMax = indicesMaxLista(qtds)
    if len(indicesMax)==1:
        df.iloc[i,colindex]=aplicFIM[indicesMax[0]]
    else:
        df.iloc[i,colindex]=str([aplicFIM[x] for x in indicesMax])
```

In [51]:

```
# Registros ainda não definidos (lista de opções)
# O filtro definirá True se APLICACAOFIM iniciar com '[' ou False caso contrário.
filtro=[]
for i, aplicacao in enumerate(df[ 'APLICACAOFIM']):
    if aplicacao[0]=='[':
        filtro.append(True)
    else:
        filtro.append(False)
```

In [52]:

```
print(f'Registros a classificar: {df[filtro].shape[0]}')
```

Registros a classificar: 67

Precisamos agora fazer a observação manual dos registros divergentes para correção. A seguir, exportaremos o arquivo em excel para fazer a classificação manual em outro notebook.

Exportando o DataSet para classificação "manual"

Exportando para um arquivo CSV

In [53]:

```
df.to_csv(r'./bases/dataframe_modelos_classificado_manual-tf.csv', index = False, header = True)
```

Exportando para um arquivo de planilha do Excel

In [54]:

```
df.to_excel(r'./bases/dataframe_modelos_classificado_manual-tf.xlsx', index  
= False, header = True)
```

In [55]:

```
tempotot=time.time()-initot  
if tempotot>60:  
    print(f'Tempo total de execução: {tempotot/60:.2f} minutos.')  
else:  
    print(f'Tempo total de execução: {tempotot:.2f} segundos.')
```

Tempo total de execução: 2.17 minutos.

Notebook Jupyter 6_NLP_modeloClassificador_parteManual

Classificação da Aplicação divergente de forma manual

Importando bibliotecas

In [1]:

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

In [2]:

```
# Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
```

Código executado em 20/01/2022 às 16:58

Carregando dataset

In [3]:

```
# Importa base de dados com os modelos já determinados para um dataframe
df = pd.read_excel(r'./bases/dataframe_modelos_classificado_manual-tf.xlsx')
)
df.iloc[:, -5: ].head()
```

Out[3]:

	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB	APLICACAOFIM
0	150 cg fan honda titan	HONDA CG FAN	HONDA CG FAN	HONDA CG TIT TITAN 125 150 160	HONDA CG FAN
1	125 cargo cg honda titan	HONDA CG TIT TITAN 125 150 160	HONDA CG TIT TITAN 125 150 160	HONDA CG TIT TITAN 125 150 160	HONDA CG TIT TITAN 125 150 160
2	125 cg fan honda	HONDA CG FAN	HONDA CG FAN	HONDA CG 125	HONDA CG FAN
3	biz c100 honda	HONDA BIZ C100 125 C125	HONDA BIZ C100 125 C125	HONDA BIZ C100 125 C125	HONDA BIZ C100 125 C125
4	150 kasinski mirage	KASINSKI MIRAGE 150 250	KASINSKI MIRAGE 150 250	KASINSKI MIRAGE 150 250	KASINSKI MIRAGE 150 250

In [4]:

```
# Verifica o tamnanho do dataframe
df.shape
```

Out[4]:

(17484, 9)

In [5]:

```
dfaplicacoes = pd.read_csv(r'./bases/Aplicacoes.csv')
```

Determinando os registros divergentes

In [6]:

```
# Registros ainda não definidos (Lista de opções)
# O filtro definirá True se APLICACAOFIM iniciar com '[' ou False caso contrário.
filtro=[]
for i, aplicacao in enumerate(df['APLICACAOFIM']):
    if aplicacao[0]=='[':
        filtro.append(True)
    else:
        filtro.append(False)
```

In [7]:

```
df[filtró].iloc[:, -6:]
```

Out[7]:

	DESCRICAO	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB
11	hunter max 125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
282	max 125 hunter p21b	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
304	hunter max125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
450	sundown hunter max 125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
848	hunter max125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
1351	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
1683	hunter max 125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
1751	max 125 hunter p21b	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
2043	cb twister 250 top	250 cb honda top twister	KAHENATOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
2740	hunter max125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
2880	sundown max125 max 125 sed hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
2958	hunter max 125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
3724	cb twister 250 top	250 cb honda top twister	KAHENATOP	HONDA TWISTER CBX 250	HONDA CB 250 250F

	DESCRICAQAO	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB
4370	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
4404	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
4494	max 125 hunter p21b	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
4510	cb twister 250 top	250 cb honda top twister	KAHEN TOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
4595	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
4638	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
5181	hunter max 125 15 x118	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
5274	cb twister 250 top	250 cb honda top twister	KAHEN TOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
5334	sundown hunter max 125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
5739	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
5784	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
6157	xlv 700 transalp	700 honda transalp	HONDA XL 700V TRANSALP	KAWASAKI NINJA 700	HONDA NC 700X 700
6170	xlv 700 transalp	700 honda transalp	HONDA XL 700V TRANSALP	KAWASAKI NINJA 700	HONDA NC 700X 700

	DESCRICAQAO	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB
6511	hunter max 125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
6559	hunter max125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
6634	cb twister 250 top	250 cb honda top twister	KAHENATOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
6642	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
...
7774	sundown max 125 sed hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
7829	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
7911	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
7952	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
8073	xlv 700 transalp	700 honda transalp	HONDA XL 700V TRANSALP	KAWASAKI NINJA 700	HONDA NC 700X 700
8144	max 125 hunter p21b	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
8523	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
9537	hunter max 125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
9738	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER

	DESCRICAQ	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB
11188	hunter max 125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
11261	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
11284	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
11560	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
12178	p21b max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
12286	sundown max 125 sed hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
12665	cb twister 250 top	250 cb honda top twister	KAHEN TOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
12707	cb twister 250 top	250 cb honda top twister	KAHEN TOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
13051	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
13267	max 125 hunter p21b	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
13867	max 125 hunter mot	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
13978	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
15025	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER

	DESCRICAQ	Modelo	APLICACAO	APLICACAOSVC	APLICACAOMNB
15223	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
15468	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
15519	hunter 125 max	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
15962	cb twister 250 top	250 cb honda top twister	KAHEN TOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
15977	cb twister 250 top	250 cb honda top twister	KAHEN TOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
16472	hunter max125	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
16795	hunter max 125 15 x118	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER
17180	max 125 hunter	125 hunter max sundown	XXX	SUNDOWN MAX	SUNDOWN HUNTER

67 rows × 6 columns

In [8]:

```
df['DESCRICAQ DO PRODUTO'].iloc[7]
```

Out[8]:

'80373 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE, COROA E PINHÃ O PARA MOTOCICLETA SHINERAY PHOENIX 50CC, MARCA ALLEN.'

In [9]:

```
print(f'Registros a classificar: {df[filtro].shape[0]}')
```

Registros a classificar: 67

Classificando os registros divergentes

Primeiramente, observando-se os registros verifica-se que são modelos repetidos, então faremos um dicionário com o par de escolha (modelo: aplicação).

In [10]:

```
def ordena(modelo):
    modelo=modelo.split()
    modelo.sort()
    modelo=" ".join(modelo)
    return modelo
def limpaColchetes(aplicacaofim):
    aplicacaofim=" ".join(aplicacaofim)
    aplicacaofim=aplicacaofim.replace("''", '')
    aplicacaofim=aplicacaofim.replace("[", '')
    aplicacaofim=aplicacaofim.replace("]", '')
    aplicacaofim=aplicacaofim.split(",")
    return aplicacaofim
```

In [11]:

```
lista=df[filtro].groupby('Modelo').agg(lambda x: list(set(x))).reset_index()
()[['Modelo','APLICACAOFIM']]
lista['Modelo']=lista['Modelo'].apply(ordena)
lista['APLICACAOFIM']=lista['APLICACAOFIM'].apply(limpaColchetes)
print(f'Número de registros a classificar: {df[filtro].shape[0]}')
print(f'Número de casos a analisar: {lista.shape[0]}')
```

Número de registros a classificar: 67
 Número de casos a analisar: 3

Observa-se que são pouquíssimos casos a analisar.

In [12]:

```
lista.sample(3)
```

Out[12]:

	Modelo	APLICACAOFIM
2	700 honda transalp	[HONDA XL 700V TRANSLALP, HONDA NC 700X 700]
0	125 hunter max sundown	[SUNDOWN MAX, SUNDOWN HUNTER]
1	250 cb honda top twister	[HONDA TWISTER CBX 250, HONDA CB 250 250F]

In [13]:

```
lista
```

Out[13]:

	Modelo	APLICACAOFIM
0	125 hunter max sundown	[SUNDOWN MAX, SUNDOWN HUNTER]
1	250 cb honda top twister	[HONDA TWISTER CBX 250, HONDA CB 250 250F]
2	700 honda transalp	[HONDA XL 700V TRANSALP, HONDA NC 700X 700]

In [14]:

```
# com essa Lista bastará deixar a opção correta.
relacoes={}
for i, x in enumerate(lista['Modelo']):
    relacoes[x]=lista['APLICACAOFIM'][i]
```

In [15]:

```
relacoes
```

Out[15]:

```
{'125 hunter max sundown': ['SUNDOWN MAX', ' SUNDOWN HUNTER'],
 '250 cb honda top twister': ['HONDA TWISTER CBX 250', ' HONDA
CB 250 250F'],
 '700 honda transalp': ['HONDA XL 700V TRANSALP', ' HONDA NC 7
00X 700']}
```

In [16]:

```
print(f'Número de relações a identificar manualmente: {len(relacoes)})')
```

Número de relações a identificar manualmente: 3

Identificadas as relações construiu-se o dicionário abaixo para classificar de acordo com sua definição de chave-valor.

In [17]:

```
# Escolhas das relações realizada manualmente (somente 26 - deixamos uma de
fora de propósito.)
relacoes = {'125 hunter max sundown': ' SUNDOWN HUNTER',
            '250 cb honda top twister': 'HONDA TWISTER CBX 250',
            '700 honda transalp': 'HONDA XL 700V TRANSALP'}
```

In [18]:

```
colindex = df.columns.get_loc("APLICACAOFIM")
for i, row in df.iterrows():
    if row['APLICACAOFIM'][0]=='[':
        try:
            modelo=row['Modelo'].split()
            modelo.sort()
            modelo=" ".join(modelo)
            relacoes[modelo]
        except:
            continue
    df.iloc[i,colindex]=relacoes[modelo]
```

Reaplicando o filtro

In [19]:

```
# Registros ainda não definidos (lista de opções)
# O filtro definirá True se APLICACAOFIM iniciar com '[' ou False caso contrário.
filtro=[]
for i, aplicacao in enumerate(df['APLICACAOFIM']):
    if aplicacao[0]=='[':
        filtro.append(True)
    else:
        filtro.append(False)
```

In [20]:

```
print(f'Registros a classificar: {df[filtro].shape[0]}')
```

Registros a classificar: 0

Precisamos agora fazer a observação manual dos registros divergentes para correção. A seguir, para os registros que não estiverem no diconário, cada linha apresentará a escolha manual de qual das opções deverá ser a APLICACAOFIM.

In [21]:

```
# As escolhas serão em um dicionário para poder retomar
# {index: escolha} ==> exemplo: {11: 1, 16: 1}
escolhas={}
from pathlib import Path
fileName = r"./pickle/escolhas.pkl"
fileObj = Path(fileName)
# se o arquivo existir
if fileObj.is_file():
    with open(fileName, 'rb') as file:
        escolhas = pickle.load(file)
        file.close()
else:
    escolhas={}
```

In [22]:

```
len(escolhas.keys())
```

Out[22]:

```
0
```

In [23]:

```
# colindex = df[filtro].columns.get_loc("APLICACAOFIM")
for i, row in df[filtro].iterrows():
    print(i)
    ultimo_i=max(escolhas.keys()) if len(escolhas.keys())>0 else 0
    a=0
    if i>100000 or a=='X': # serve para cancelar e estabelecer um limite para as verificações até o índice i
        break
    if row['APLICACAOFIM'][0]=='[':
        # guarda a lista das opções na variável
        aplictemp=row['APLICACAOFIM'].replace("[","").replace(", ",";").replace(",;").replace("","","").replace("]", "")
        aplictemp=aplictemp.split(';')
        print('*'*20)
        print('Digite o número de uma das opções e tecle Enter:')
        print('Modelo: ', row['Modelo'])
        for n, aplicacao in enumerate(aplictemp):
            aplictemp[n]=aplicacao.strip()
            print(' ',str(n+1)+')',aplictemp[n])
        pedeEscolha=False
        if ultimo_i>=i:
            try:
                a=escolhas[i]
            except:
                pedeEscolha=True
        if pedeEscolha or ultimo_i<i:
            while True:
                try:
                    a=input('Escolha uma das opções acima, 0 para outra ou X para cancelar: ').upper()
                    if a=='X':
                        break
                    elif a=='0' or a=='o':
                        outra=input('Digite o nome da aplicação:').upper()
                        # verificar se está em aplicações
                        if outra in dfaplicacoes['APLICACOES'].tolist():
                            df.iloc[i,colindex]=outra # define a aplicação corrente
                            break # encerra o while
                        a=int(a)
                        if a not in range(1,len(aplictemp)+1):
                            raise(ValueError)
                except ValueError:
                    print("\nOpção inválida.\nDigite o número de uma das opções apresentadas:")
                else:
                    break # encerra o while
            if a=='X': break # se a entrada for X encerra o for
            if a=='0': continue # se a entrada for 0 segue para o próximo
        print(aplictemp[a-1],'\n\n')
        df.iloc[i,colindex]=aplictemp[a-1]
        escolhas[i]=a # acrescenta ou atualiza a escolha no dicionário
#np.save('escolhas.npy', escolhas) # salva o dicionário em um arquivo pickle
```

```

with open(r'./pickle/escolhas.pkl', 'wb') as file:
    pickle.dump(escolhas, file)
    file.close
print(escolhas)

{}

```

Depois da análise manual todos os registros estão classificados para podermos utilizar como base para treinar o nosso classificador definitivo.

In [24]:

```

# Registros ainda não definidos (Lista de opções)
# O filtro definirá True se APLICACAOFIM iniciar com '[' ou False caso contrário.
filtro=[]
for i, aplicacao in enumerate(df['APLICACAOFIM']):
    if aplicacao[0]=='[':
        filtro.append(True)
    else:
        filtro.append(False)

```

In [25]:

```
df[filtro]
```

Out[25]:

PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO	Modelo	APLICACAO

In [26]:

```
print(f'Registros a classificar: {df[filtro].shape[0]}')
```

Registros a classificar: 0

Comparando as classificações

In [27]:

```

# Definição dos filtros
f1 = df['APLICACAOsvc'] == df['APLICACAOFIM'] # Modelo manual e SVC iguais
f2 = df['APLICACAOmnb'] == df['APLICACAOFIM'] # Modelo manual e MNB iguais

```

In [28]:

```
# Quantidade de registros divergentes entre o modelo SVC e o modelo final
df[~f1].shape
```

Out[28]:

(898, 9)

In [29]:

```
# Quantidade de registros divergentes entre o modelo MNB e o modelo final
df[~f2].shape
```

Out[29]:

(2954, 9)

Eliminando as colunas de classificação

In [30]:

```
df=df.assign(APLICACAO=df.APLICACAOFIM.tolist())
```

In [31]:

```
df = df.drop('APLICACAOSVC', 1)
df = df.drop('APLICACAOMNB', 1)
df = df.drop('APLICACAOFIM', 1)
```

In [32]:

```
df.iloc[0:,-3:].sample(5)
```

Out[32]:

	DESCRICAO	Modelo	APLICACAO
16685	titan 160 fan	160 fan honda titan	HONDA CG FAN
12959	cg 160 titan fan start cargo	160 cargo cg fan honda titan	HONDA CG FAN
13313	cg 160 titan fan start cargo	160 cargo cg fan honda titan	HONDA CG FAN
4430	broz 160	160 broz honda	HONDA NXR 150 160 BROZ
7245	fazer ys 250	250 fazer yamaha	YAMAHA FAZER YS250 250

Criando a coluna de existência de corrente com Retentor no Kit

Diante da possibilidade do kit de transmissão vir acompanhado ou não de corrente com retentor, e esta questão influenciar no preço do produto, faz-se necessário criar uma coluna do tipo *boolean* para indicar ou não a presença de corrente com retentor no kit.

Após a análise do dataset, observou-se que todos os kits que possuíam corrente com retentor havia na descrição uma das seguintes opções:

- com retentor
- c/retentor
- c/ retentor
- com ret
- c/ret
- c/ ret

Desse modo, definiu-se o padrão Regex para encontrar essas formas na descrição e colocar True|False na coluna RETENTOR

Função que determina a existência de retentor na corrente do kit e retorna True|False

In [33]:

```
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
```

In [34]:

```
# cria a coluna RETENTOR com a indicação True/False
df['RETENTOR']=df['DESCRICAO DO PRODUTO'].apply(retentorAux)
```

Exportando o DataSet Classificado

Exportando para um arquivo CSV

In [35]:

```
df.to_csv(r'./bases/dataframe_modelos_classificado.csv', index = False, header = True)
```

Exportando para um arquivo de planilha do Excel

In [36]:

```
df.to_excel(r'./bases/dataframe_modelos_classificado.xlsx', index = False, header = True)
```

In [37]:

```
tempotot=time.time()-initot
if tempotot>60:
    print(f'Tempo total de execução: {tempotot/60:.2f} minutos.')
else:
    print(f'Tempo total de execução: {tempotot:.2f} segundos.')
```

Tempo total de execução: 5.47 segundos.

Notebook Jupyter 7_treinamentoClassificador

Classificação da Aplicação por aprendizado de máquina

Importando bibliotecas

In [1]:

```
import pandas as pd, time
# Vetorização
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
# Modelos
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
# Métricas
from sklearn import metrics
# Divisor de Treino/Teste
from sklearn.model_selection import train_test_split
# Matplot
import matplotlib.pyplot as plt
```

In [2]:

```
# importa funções criadas no TCC e que precisam ser reutilizadas
from funcoesTCC import *
# Funções: criaModelo, LimpaDescricao, achaPalavraChave, pegaChave, acrescentaMarca, retentorAux, classificaAplicacaoSVC
# Variáveis: stopwords, palavrasChave, Marcas, cvt, tfi, clfsvc
# Datasets: dfaplicacoes (Aplicações)
```

In [3]:

```
# Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
```

Código executado em 20/01/2022 às 17:23

Carregando dataset

In [4]:

```
# Importa base de dados com os modelos já determinados para um dataframe
df = pd.read_excel('bases/dataframe_modelos_classificado.xlsx')
df.iloc[2100:-3].head()
```

Out[4]:

	Modelo	APLICACAO	RETENTOR
2100	160 broz honda nxr xr xre	HONDA NXR 150 160 BROZ	False
2101	150 broz honda nxr xr	HONDA NXR 150 160 BROZ	False
2102	125 xlr	HONDA XLR	False
2103	biz c100 honda	HONDA BIZ C100 125 C125	False
2104	biz c100 honda	HONDA BIZ C100 125 C125	False

In [5]:

```
df.sample(5)
```

Out[5]:

	PAIS DE ORIGEM	DESCRICAO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAO	Mode
10427	CHINA, REPUBLICA POP	Ý10822/45" KIT DE TRANSMISSÃO, EM AÇO 1045, C...	3.1500	i45" biz	b honc
8644	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO P/ MOTOCICLETA MOD.: CG 125...	3.6100	cg 125 cargo fan	12 carç cg fá honc
12511	CHINA, REPUBLICA POP	item 22;Partes e peças para Motocicletas,Desta...	4.9450	ybr 125 factor	12 fact yamaç y
6179	CHINA, REPUBLICA POP	22376 - 71899 - KIT DE TRANMISSAO PARA MOTOCI...	4.2738	pop	honc pc
652	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO , MARCA RIFFEL, TITANIUM (1...	3.9372	biz 125	125 b honc

In [6]:

```
# Verifica o tamnanho do dataframe
df.shape
```

Out[6]:

(17484, 7)

Define linha de exemplo

In [7]:

```
linha=15 # Linha a ser utilizada como exemplo
descricao = df.iloc[linha]['DESCRICAO DO PRODUTO']
# verifica a descrição do produto
descricao
```

Out[7]:

'80372 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE, COROA E PINHÃ O PARA MOTOCICLETA RIVA150 DAFRA, MARCA ALLEN.'

Exemplo da função que cria o modelo a ser utilizado no classificador treinado

In [8]:

```
# cria a descrição do modelo que será usado na predição
criaModelo(descricao)
```

Out[8]:

'150 dafra riva'

CountVectorizer

CountVectorizer do DataSet

In [9]:

```
# Criação da função CountVectorizer
cvt = CountVectorizer(strip_accents='ascii', lowercase=True)
X_cvt = cvt.fit_transform(df['Modelo'])
```

Um ponto muito importante desse trabalho foi a decisão quanto à **utilização ou não do TF-IDF** (*term frequency-inverse document frequency*), que serve para indicar o grau de importância de uma palavra em relação ao conjunto total de palavras existentes. Na prática, funciona com um ponderador dos termos na classificação.

Especificamente no nosso trabalho, o objetivo é determinar a aplicação a que se refere uma descrição, então a existência de um termo mais raro, que ganharia peso com o TF-IDF, não pode se sobrepor à aparição de vários termos que indicam uma classificação.

Por exemplo, dentro da construção dos códigos, com a utilização da ponderação com o TF-IDF, a descrição já realizada a limpeza “yamaha fazer 150 gt” era classificada pelo modelo *Linear SVC* como “SUZUKI GT”, mas a aplicação mais adequada seria “YAMAHA FAZER YS150 150”.

Decidiu-se, então, pela não utilização do ponderador TF-IDF.

In [10]:

```
# Criação da função TfidfTransformer
tfi = TfidfTransformer(use_idf=True)
X_tfi = tfi.fit_transform(X_cvt)
```

In [11]:

```
# A entrada serão os vetores do CountVectorizer
entrada = X_cvt
# A saída será as aplicações
saída = df['APLICACAO']
# Separando 30% dos dados para teste
X_train, X_test, y_train, y_test = train_test_split(entrada, saída, test_size=0.3)
```

Treinando os Modelos

Modelo LinearSVC

Treinamento do modelo

In [12]:

```
# Criando modelo
clfsvc = LinearSVC()
# Treinamento do modelo
clfsvc.fit(X_train, y_train)
```

Out[12]:

```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
intercept_scaling=1, loss='squared_hinge', max_iter=1000,
multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
verbose=0)
```

Função de classificação LinearSVC

A função para utilização do modelo, recebe a descrição filtrada Modelo e retorna a aplicação.

In [13]:

```
def classificaAplicacaoSVC(modelo):
    novo_cvt = cvt.transform(pd.Series(modelo))
    aplicacao = clfsvc.predict(novo_cvt)[0]
    return aplicacao
```

No final o nosso resultado mostrando (Modelo: descrição filtrada para o modelo e Aplicação: aplicação prevista).

In [14]:

```
# Lista de exemplos de novos produtos
modelos = ['150 CG HONDA TITAN',
           '125 CARGO CG HONDA TITAN',
           'BIZ C100 HONDA',
           '100 HONDA BIZ',
           '100 BIZ BRAVO HONDA',
           '125 YBR GT YAMAHA',
           '250F TWISTER HONDA']
# Loop for para fazer a predição do departamento de novos produtos
for modelo in modelos:
    print('Modelo:', modelo, 'Aplicação:', classificaAplicacaoSVC(modelo))
```

```
Modelo: 150 CG HONDA TITAN      Aplicação: HONDA CG TIT TITAN
125 150 160
Modelo: 125 CARGO CG HONDA TITAN      Aplicação: HONDA CG TIT
TITAN 125 150 160
Modelo: BIZ C100 HONDA      Aplicação: HONDA BIZ C100 125 C12
5
Modelo: 100 HONDA BIZ      Aplicação: HONDA BIZ C100 125 C125
Modelo: 100 BIZ BRAVO HONDA      Aplicação: HONDA BIZ C100 12
5 C125
Modelo: 125 YBR GT YAMAHA      Aplicação: YAMAHA FACTOR YBR 1
25 YBR125
Modelo: 250F TWISTER HONDA      Aplicação: HONDA HERO
```

Predição e avaliação do Modelo Linear SVC

Métricas

In [15]:

```
# tempo de execução para 10.000 classificações
ini=time.time()
#teste=df.sample(10000)['Modelo'].apply(classificaAplicacaoSVC)
teste=df.sample(100)['Modelo'].apply(classificaAplicacaoSVC)
fim=time.time()
```

In [16]:

```
# Realizando a predição
resultadosvc = clfsvc.predict(X_test)
# Avaliando o modelo
print('Acurácia: {:.2f}'.format(metrics.accuracy_score(y_test, resultadosvc)))
print('Precisão: {:.2f}'.format(metrics.precision_score(y_test, resultadosvc,average='micro')))
print('Recall: {:.2f}'.format(metrics.recall_score(y_test, resultadosvc,average='micro')))
print('F1_Score: {:.2f}'.format(metrics.f1_score(y_test, resultadosvc,average='micro')))
print("Tempo: " + str(round((fim-ini),1)) + " segundos.")
```

Acurácia: 1.00
Precisão: 1.00
Recall: 1.00
F1_Score: 1.00
Tempo: 0.0 segundos.

In [17]:

```
# Avaliação completa
print(metrics.classification_report(y_test, resultadosvc))
```

support		precision	recall	f1-score
	SUNDOWN HUNTER	1.00	1.00	1.00
23	BMW F800GS	1.00	1.00	1.00
2	BRAVAX BVX STREET 130	1.00	0.67	0.80
3	DAFRA APACHE	1.00	1.00	1.00
9	DAFRA KANSAS	1.00	1.00	1.00
16	DAFRA NEXT	1.00	1.00	1.00
8	DAFRA RIVA	1.00	1.00	1.00
8	DAFRA SPEED	1.00	1.00	1.00
12	DAFRA SUPER	0.86	1.00	0.92
6	DAFRA ZIG	1.00	1.00	1.00
4	HONDA BIZ C100 125 C125	1.00	1.00	1.00
491	HONDA CB 250 250F	1.00	1.00	1.00
8	HONDA CB 300R 300 CB300	1.00	1.00	1.00
149	HONDA CB 500	1.00	1.00	1.00
1	HONDA CB HORNET 600	1.00	1.00	1.00
6	HONDA CBX 1000	1.00	1.00	1.00
4	HONDA CG 125	1.00	1.00	1.00
33	HONDA CG FAN	1.00	1.00	1.00
887	HONDA CG TIT TITAN 125 150 160	0.99	0.99	0.99
483	HONDA CG TODAY	1.00	0.86	0.92
7	HONDA CRF 230 230F 250 250F	1.00	1.00	1.00
36	HONDA DREAM	1.00	1.00	1.00
19	HONDA NC 700X 700	1.00	1.00	1.00
1	HONDA NX 150 200 250	1.00	0.90	0.95
10	HONDA NX 350 SAHARA	1.00	1.00	1.00
4	HONDA NX 400 FALCON	1.00	1.00	1.00
39	HONDA NXR 150 160 BROZ	1.00	1.00	1.00

628	HONDA POP	1.00	1.00	1.00
247	HONDA STRADA CBX 200	1.00	1.00	1.00
43	HONDA TORNADO XR 250	0.99	1.00	0.99
81	HONDA TWISTER CBX 250	1.00	1.00	1.00
185	HONDA XL 700V TRANSALP	1.00	1.00	1.00
1	HONDA XL XLS 125 XL125 XL125S	1.00	1.00	1.00
60	HONDA XLR	1.00	1.00	1.00
61	HONDA XR	1.00	0.93	0.97
15	HONDA XRE 300	1.00	1.00	1.00
159	KAHENNA TOP	0.83	0.83	0.83
6	KASINSKI COMET GT 150	1.00	1.00	1.00
1	KASINSKI COMET GT 250	1.00	1.00	1.00
1	KASINSKI MIRAGE 150 250	1.00	1.00	1.00
4	KAWASAKI ER-6N ER6N ER-6F ER6F	1.00	1.00	1.00
2	KAWASAKI MAXI	1.00	0.67	0.80
3	KAWASAKI NINJA 250 300	1.00	1.00	1.00
17	KTM EXC 500	0.00	0.00	0.00
1	KTM SX 125	1.00	1.00	1.00
1	KTM SX 150	1.00	1.00	1.00
11	KTM SX 250	0.00	0.00	0.00
3	KTM SX 50	1.00	1.00	1.00
2	KTM XC 525	1.00	1.00	1.00
1	MRX 230R 230	1.00	1.00	1.00
1	MVK MA	1.00	0.67	0.80
3	MVK SPORT	0.00	0.00	0.00
1	SHINERAY JET 50	0.80	0.80	0.80
5	SHINERAY LIBERTY 50	1.00	1.00	1.00
1	SHINERAY PHOENIX 50	0.94	1.00	0.97
16				

4	SHINERAY XY 250	1.00	1.00	1.00
6	SUNDOWN HUNTER	1.00	1.00	1.00
1	SUNDOWN MAX	1.00	1.00	1.00
1	SUNDOWN STX MOTARD	1.00	1.00	1.00
1	SUNDOWN WEB	0.99	0.99	0.99
69	SUZUKI BANDIT GSF 750	1.00	1.00	1.00
1	SUZUKI GS 500	1.00	1.00	1.00
6	SUZUKI GSR	1.00	1.00	1.00
5	SUZUKI INTRUDER	1.00	0.75	0.86
4	SUZUKI KATANA	1.00	0.97	0.98
33	SUZUKI V-STROM DL STROM	1.00	1.00	1.00
1	SUZUKI YES EN 125	0.98	1.00	0.99
60	TRAXX STAR 50	1.00	1.00	1.00
2	TRIUMPH TIGER	1.00	1.00	1.00
1	YAMAHA CRYPTON	1.00	1.00	1.00
72	YAMAHA FACTOR YBR 125 YBR125	1.00	1.00	1.00
385	YAMAHA FAZER FZ6	1.00	1.00	1.00
2	YAMAHA FAZER YS150 150	0.99	1.00	1.00
151	YAMAHA FAZER YS250 250	0.99	1.00	1.00
194	YAMAHA LANDER XTZ 250	1.00	1.00	1.00
124	YAMAHA XJ6	1.00	1.00	1.00
2	YAMAHA XT 225	1.00	1.00	1.00
3	YAMAHA XT 600	1.00	1.00	1.00
3	YAMAHA XT 660R 660	1.00	1.00	1.00
7	YAMAHA XTZ 125	0.99	1.00	1.00
123	YAMAHA XTZ CROSSER 150	1.00	1.00	1.00
123	YAMAHA XTZ TENERE 250	0.89	1.00	0.94
25	YAMAHA YZF R1	1.00	1.00	1.00
9				

	micro avg	1.00	1.00	1.00
5246	macro avg	0.95	0.94	0.95
5246	weighted avg	1.00	1.00	1.00
5246				

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
```

```
'precision', 'predicted', average, warn_for)
```

O alerta ao final do relatório é esperado, tendo em vista que existem aplicações raras e que não aparecerem no grupo de teste.

Matriz de Confusão

In [18]:

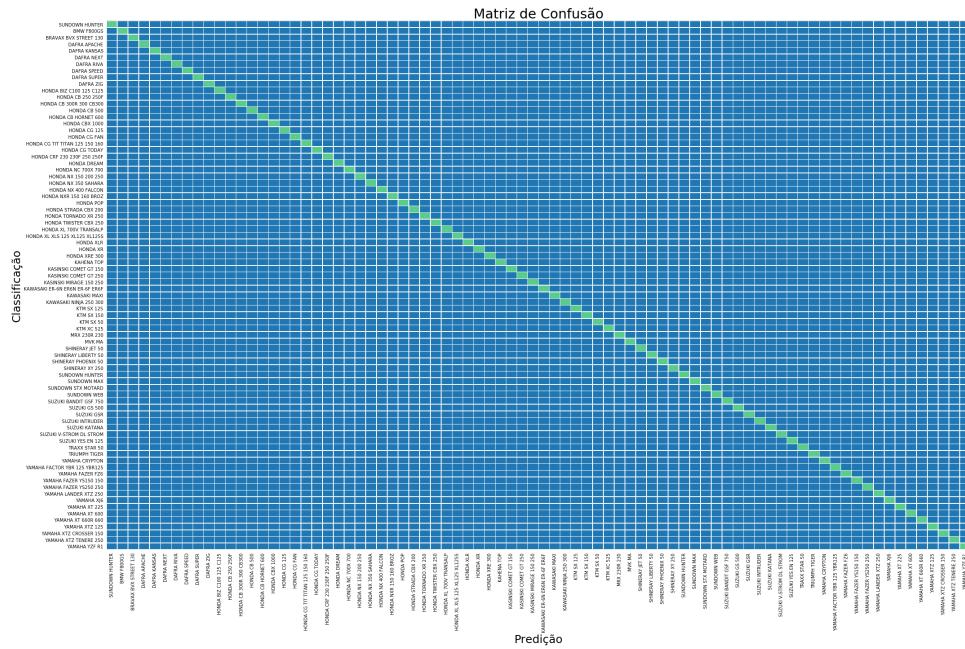
```
# importa os módulos
# gera a matriz de confusão formatada (adaptado por mim)
from prettyPlotConfusionMatrix import pretty_plot_confusion_matrix # pretty
PlotConfusionMatrix.py local
# código disponível em https://github.com/wcipriano/pretty-print-confusion-
matrix
from sklearn.metrics import confusion_matrix
```

In [19]:

```
aplicacoessvc = np.unique(resultadosvc) # define as aplicações presentes no
resultado
cmsvc = confusion_matrix(y_test, resultadosvc, aplicacoessvc) # cria a matriz de confusão
dfmcsvc=pd.DataFrame(cmsvc,index=aplicacoessvc,columns=aplicacoessvc) # converte a matriz em dataframe
```

In [20]:

```
pretty_plot_confusion_matrix(dfmcsvc, annot=True, cmap="tab10", fmt=".0f",
fz=1, lw=0.5, cbar=False,
figsize=[30,20], show_null_values=2, pred_val_
axis='x',insertTot=False)
```



Apesar do grande número de classes, pode-se observar a linha formada pelos acertos na diagonal da imagem, demonstrando o alto índice de acerto do modelo.

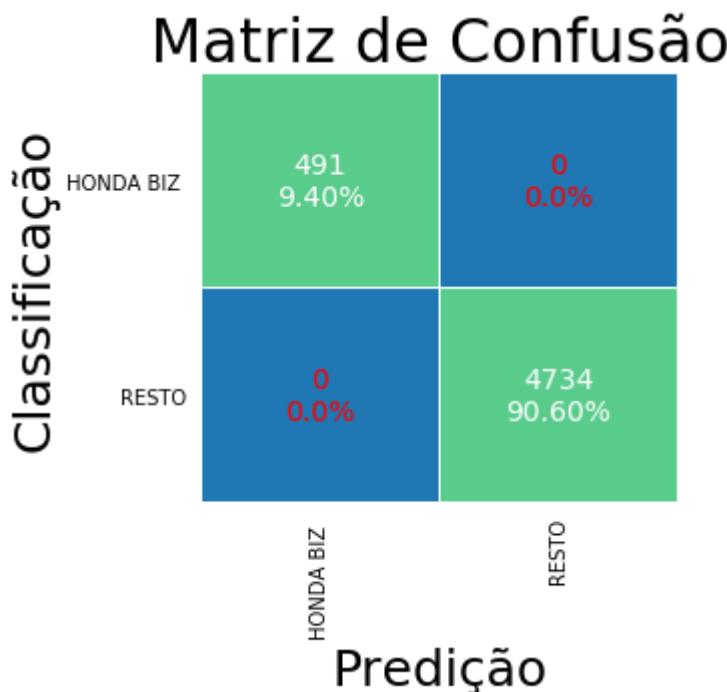
Para melhorar a visão, esboçaremos a matriz de confusão para o item 'HONDA BIZ 100 C100 125 C125' versus o resto (OvR), isso nos permitirá ver a classificação como se fosse uma classificação binária OvR do tipo ou é a classificação demonstrada ou é o resto.

In [21]:

```
# Função que plota a matriz de confusão "one versus rest (OvR)" do modelo
def plotConfusaoOvR(modelo, predicao, cm):
    # modelo = modelo de aplicação que será o "one" no "one versus rest"
    # predicao = resultado da predição do modelo
    # cm = matriz de confusão gerada do modelo
    aplicacoes = np.unique(predicao) # define os aplicações presentes no resultado
    pos = int(np.where(aplicacoes == modelo)[0]) # econtra a posição de modelo nas aplicações
    TP=cm[pos][pos] # True Positive
    FP=sum([cm[pos][x] for x in range(cm.shape[0])])-TP # False Positive
    FN=sum([cm[x][pos] for x in range(cm.shape[0])])-TP # False Negative
    TN = sum([cm[x][x] for x in range(cm.shape[0])])-TP # True Negative
    confmod=np.array([[TP,FP],[FN,TN]]) # gera a matriz de confusão
    labels = [modteste[:10], 'RESTO'] # reduz o nome do modelo aos primeiros 10 caracteres
    pretty_plot_confusion_matrix(pd.DataFrame(confmod, index=labels, columns=labels), annot=True, cmap="tab10",
                                  fmt='.2f', fz=14, lw=0.5, cbar=False, figsize=[5,5], show_null_values=2,
                                  pred_val_axis='x', insertTot=False)
```

In [22]:

```
# gera o plot da matriz de confusão OvR para o modtste do resultadosvc
modteste=dfaplicacoes['APLICACOES'].iloc[276]
plotConfusaoOvR(modteste, resultadosvc, cmsvc)
```



Utilização do Modelo

O modelo deverá receber uma descrição do produto conforme entrada do contribuinte na Declaração de Importação e deverá retornar a classificação de aplicação da motocicleta.

Teste com a linha de exemplo

In [23]:

```
print('index:', linha)
descricao = df.iloc[linha]['DESCRICAO DO PRODUTO']
# verifica a descrição do produto
descricao
```

index: 15

Out[23]:

'80372 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE, COROA E PINHÃ O PARA MOTOCICLETA RIVA150 DAFRA, MARCA ALLEN.'

In [24]:

```
classificaAplicacaoSVC(criaModelo(descricao))
```

Out[24]:

'DAFRA RIVA'

Modelo Multinomial Naive Bayes

Treinamento do modelo

In [25]:

```
# Criando modelo
clfMNB = MultinomialNB()
# Treinamento do modelo
clfMNB.fit(X_train, y_train)
```

Out[25]:

MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

Função de classificação

A função para utilização do modelo, recebe a descrição filtrada Modelo e retorna a aplicação.

In [26]:

```
def classificaAplicacaoMNB(modelo):
    novo_cvt = cvt.transform(pd.Series(modelo))
    aplicacao = clfmnb.predict(novo_cvt)[0]
    return aplicacao
```

No final o nosso resultado mostrando (Modelo: descrição filtrada para o modelo e Aplicação: aplicação prevista).

In [27]:

```
# Lista de exemplos de novos produtos
modelos = ['150 CG FAN HONDA TITAN',
            '125 CARGO CG HONDA TITAN',
            'BIZ C100 HONDA',
            '100 HONDA BIZ',
            '100 BIZ BRAVO HONDA',
            '125 YBR GT YAMAHA',
            '250F TWISTER HONDA']
# Loop for para fazer a predição do departamento de novos produtos
for modelo in modelos:
    print('Modelo:', modelo, 'Aplicação:', classificaAplicacaoMNB(modelo))
```

Modelo: 150 CG FAN HONDA TITAN Aplicação: HONDA CG FAN
 Modelo: 125 CARGO CG HONDA TITAN Aplicação: HONDA CG FAN
 Modelo: BIZ C100 HONDA Aplicação: HONDA BIZ C100 125 C125
 Modelo: 100 HONDA BIZ Aplicação: HONDA BIZ C100 125 C125
 Modelo: 100 BIZ BRAVO HONDA Aplicação: HONDA BIZ C100 125 C125
 Modelo: 125 YBR GT YAMAHA Aplicação: YAMAHA FACTOR YBR 125 YBR
 125
 Modelo: 250F TWISTER HONDA Aplicação: HONDA CB 250 250F

Predição e avaliação do Modelo Multinomial Naive Bayes

Métricas

In [28]:

```
# tempo de execução para 10.000 classificações
ini=time.time()
#teste=df.sample(10000)['Modelo'].apply(classificaAplicacaoMNB)
teste=df.sample(100)['Modelo'].apply(classificaAplicacaoMNB)
fim=time.time()
```

In [29]:

```
# Realizando a predição
resultadomnb = clfmnb.predict(X_test)
# Avaliando o modelo
print('Acurácia: {:.2f}'.format(metrics.accuracy_score(y_test, resultadomnb)))
print('Precisão: {:.2f}'.format(metrics.precision_score(y_test, resultadomnb,average='micro')))
print('Recall: {:.2f}'.format(metrics.recall_score(y_test, resultadomnb,average='micro')))
print('F1_Score: {:.2f}'.format(metrics.f1_score(y_test, resultadomnb,average='micro')))
print("Tempo: " + str(round((fim-ini),1)) + " segundos.")
```

Acurácia: 0.95
Precisão: 0.95
Recall: 0.95
F1_Score: 0.95
Tempo: 0.0 segundos.

In [30]:

```
# Avaliação completa
print(metrics.classification_report(y_test, resultadomnb))
```

support		precision	recall	f1-score
	SUNDOWN HUNTER	0.77	1.00	0.87
23	BMW F800GS	1.00	1.00	1.00
2	BRAVAX BVX STREET 130	0.00	0.00	0.00
3	DAFRA APACHE	0.90	1.00	0.95
9	DAFRA KANSAS	1.00	1.00	1.00
16	DAFRA NEXT	1.00	1.00	1.00
8	DAFRA RIVA	1.00	1.00	1.00
8	DAFRA SPEED	1.00	1.00	1.00
12	DAFRA SUPER	1.00	0.83	0.91
6	DAFRA ZIG	1.00	1.00	1.00
4	HONDA BIZ C100 125 C125	1.00	1.00	1.00
491	HONDA CB 250 250F	1.00	0.88	0.93
8	HONDA CB 300R 300 CB300	0.99	1.00	0.99
149	HONDA CB 500	0.00	0.00	0.00
1	HONDA CB HORNET 600	1.00	0.67	0.80
6	HONDA CBX 1000	0.00	0.00	0.00
4	HONDA CG 125	1.00	0.06	0.11
33	HONDA CG FAN	0.92	1.00	0.96
887	HONDA CG TIT TITAN 125 150 160	0.98	0.89	0.93
483	HONDA CG TODAY	0.00	0.00	0.00
7	HONDA CRF 230 230F 250 250F	0.97	1.00	0.99
36	HONDA DREAM	1.00	0.79	0.88
19	HONDA NC 700X 700	0.00	0.00	0.00
1	HONDA NX 150 200 250	1.00	0.30	0.46
10	HONDA NX 350 SAHARA	1.00	0.75	0.86
4	HONDA NX 400 FALCON	0.97	1.00	0.99
39	HONDA NXR 150 160 BROZ	0.96	1.00	0.98

628	HONDA POP	0.98	1.00	0.99
247	HONDA STRADA CBX 200	0.86	1.00	0.92
43	HONDA TORNADO XR 250	1.00	1.00	1.00
81	HONDA TWISTER CBX 250	0.97	1.00	0.99
185	HONDA XL 700V TRANSALP	0.00	0.00	0.00
1	HONDA XL XLS 125 XL125 XL125S	1.00	1.00	1.00
60	HONDA XLR	1.00	1.00	1.00
61	HONDA XR	0.00	0.00	0.00
15	HONDA XRE 300	1.00	1.00	1.00
159	KAHENNA TOP	0.00	0.00	0.00
6	KASINSKI COMET GT 150	0.00	0.00	0.00
1	KASINSKI COMET GT 250	1.00	1.00	1.00
1	KASINSKI MIRAGE 150 250	1.00	1.00	1.00
4	KAWASAKI ER-6N ER6N ER-6F ER6F	0.00	0.00	0.00
2	KAWASAKI MAXI	0.00	0.00	0.00
3	KAWASAKI NINJA 250 300	0.89	1.00	0.94
17	KTM EXC 500	0.00	0.00	0.00
1	KTM SX 125	0.00	0.00	0.00
1	KTM SX 150	0.00	0.00	0.00
11	KTM SX 250	0.00	0.00	0.00
3	KTM SX 50	0.00	0.00	0.00
2	KTM XC 525	1.00	1.00	1.00
1	MRX 230R 230	0.00	0.00	0.00
1	MVK MA	0.00	0.00	0.00
3	MVK SPORT	0.00	0.00	0.00
1	SHINERAY JET 50	1.00	0.80	0.89
5	SHINERAY LIBERTY 50	0.00	0.00	0.00
1	SHINERAY PHOENIX 50	0.93	0.88	0.90
16				

	SHINERAY XY 250	0.00	0.00	0.00
4	SUNDOWN HUNTER	0.00	0.00	0.00
6	SUNDOWN MAX	0.00	0.00	0.00
1	SUNDOWN STX MOTARD	0.00	0.00	0.00
1	SUNDOWN WEB	0.83	1.00	0.91
69	SUZUKI BANDIT GSF 750	1.00	1.00	1.00
1	SUZUKI GS 500	1.00	1.00	1.00
6	SUZUKI GSR	1.00	1.00	1.00
5	SUZUKI INTRUDER	0.00	0.00	0.00
4	SUZUKI KATANA	1.00	0.03	0.06
33	SUZUKI V-STROM DL STROM	0.00	0.00	0.00
1	SUZUKI YES EN 125	0.62	1.00	0.77
60	TRAXX STAR 50	1.00	1.00	1.00
2	TRIUMPH TIGER	1.00	1.00	1.00
1	YAMAHA CRYPTON	1.00	1.00	1.00
72	YAMAHA FACTOR YBR 125 YBR125	1.00	1.00	1.00
385	YAMAHA FAZER FZ6	0.00	0.00	0.00
2	YAMAHA FAZER YS150 150	0.99	1.00	1.00
151	YAMAHA FAZER YS250 250	0.95	1.00	0.97
194	YAMAHA LANDER XTZ 250	0.86	1.00	0.92
124	YAMAHA XJ6	1.00	1.00	1.00
2	YAMAHA XT 225	0.00	0.00	0.00
3	YAMAHA XT 600	1.00	1.00	1.00
3	YAMAHA XT 660R 660	0.70	1.00	0.82
7	YAMAHA XTZ 125	0.98	1.00	0.99
123	YAMAHA XTZ CROSSER 150	1.00	1.00	1.00
123	YAMAHA XTZ TENERE 250	1.00	0.20	0.33
25	YAMAHA YZF R1	1.00	1.00	1.00
9				

	micro avg	0.95	0.95	0.95
5246	macro avg	0.64	0.60	0.60
5246	weighted avg	0.94	0.95	0.94
5246				

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.

```
'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
'precision', 'predicted', average, warn_for)
```

O alerta ao final do relatório é esperado, tendo em vista que existem aplicações raras e que não aparecerem no grupo de teste.

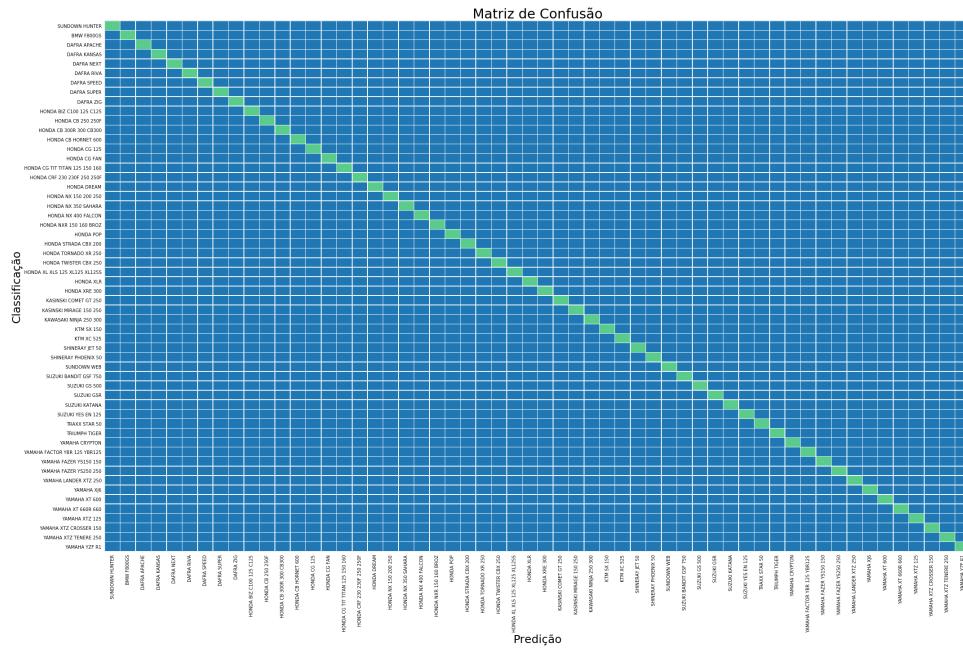
Matriz de Confusão

In [31]:

```
aplicacoesmnb = np.unique(resultadomnb) # define os aplicações presentes no resultado
cmmnb = confusion_matrix(y_test, resultadomnb, aplicacoesmnb) # cria a matriz de confusão
dfmcmmnb=pd.DataFrame(cmmnb,index=aplicacoesmnb,columns=aplicacoesmnb) # converte a matriz em dataframe
```

In [32]:

```
pretty_plot_confusion_matrix(dfmcnmb, annot=True, cmap="tab10", fmt=".0f",
fz=1, lw=0.5, cbar=False,
figsize=[30,20], show_null_values=2, pred_val_
axis='x',insertTot=False)
```

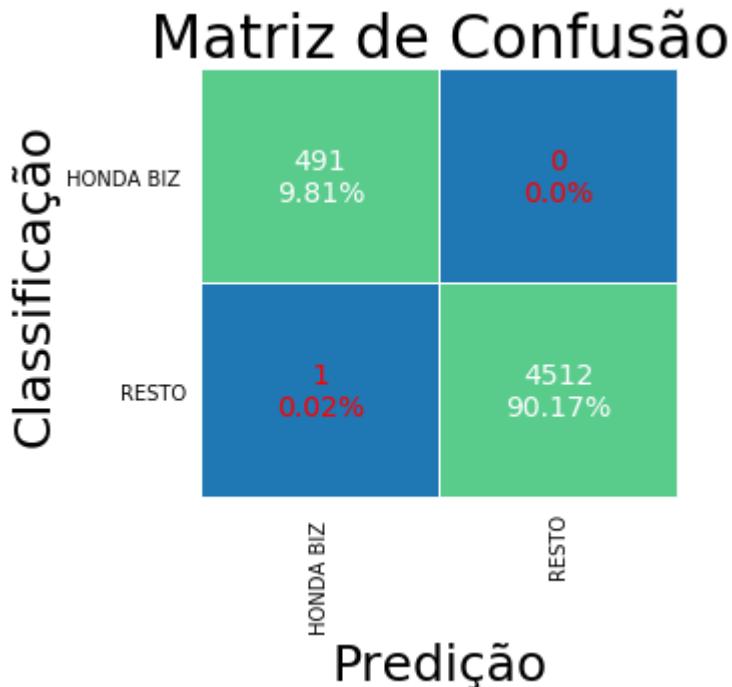


Apesar do grande número de classes, pode-se observar a linha formada pelos acertos na diagonal da imagem, demonstrando o alto índice de acerto do modelo.

Para melhorar a visão, esboçaremos a matriz de confusão para o item 'HONDA BIZ 100 C100 125 C125' versus o resto (OVR), isso nos permitirá ver a classificação como se fosse uma classificação binária OvR do tipo ou é a classificação demonstrada ou é o resto.

In [33]:

```
# gera o plot da matriz de confusão OvR para o modtste do resultadosvc
modteste=dfaplicacoes['APLICACOES'].iloc[276]
plotConfusaoOvR(modteste, resultadomnb, cmmnb)
```



Utilização do Modelo

O modelo deverá receber uma descrição do produto conforme entrada do contribuinte na Declaração de Importação e deverá retornar a classificação de aplicação da motocicleta.

Teste com a linha de exemplo

In [34]:

```
print('index:', linha)
descricao = df.iloc[linha]['DESCRICAO DO PRODUTO']
# verifica a descrição do produto
descricao
```

index: 15

Out[34]:

'80372 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE, COROA E PINHÃO PARA MOTOCICLETA RIVA150 DAFRA, MARCA ALLEN.'

In [35]:

```
classificaAplicacaoMNB(criaModelo(descricao))
```

Out[35]:

```
'DAFRA RIVA'
```

Modelo de Regressão Logística

Treinamento do modelo

In [36]:

```
# Criando modelo
clflgr = LogisticRegression(solver='lbfgs', multi_class='multinomial')
# Treinamento do modelo
clflgr.fit(X_train, y_train)
```

Out[36]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='multinomial',
                   n_jobs=None, penalty='l2', random_state=None, solver='lbfgs',
                   tol=0.0001, verbose=0, warm_start=False)
```

Função de classificação

A função para utilização do modelo, recebe a descrição filtrada Modelo e retorna a aplicação.

In [37]:

```
def classificaAplicacaoLGR(modelo):
    novo_cvt = cvt.transform(pd.Series(modelo))
    aplicacao = clflgr.predict(novo_cvt)[0]
    return aplicacao
```

No final o nosso resultado mostrando (Modelo: descrição filtrada para o modelo e Aplicação: aplicação prevista).

In [38]:

```
# Lista de exemplos de novos produtos
modelos = ['150 CG FAN HONDA TITAN',
            '125 CARGO CG HONDA TITAN',
            'BIZ C100 HONDA',
            '100 HONDA BIZ',
            '100 BIZ BRAVO HONDA',
            '125 YBR GT YAMAHA',
            '250F TWISTER HONDA']
# Loop for para fazer a predição do departamento de novos produtos
for modelo in modelos:
    print('Modelo:', modelo, 'Aplicação:', classificaAplicacaoLGR(modelo))
```

Modelo: 150 CG FAN HONDA TITAN Aplicação: HONDA CG FAN
 Modelo: 125 CARGO CG HONDA TITAN Aplicação: HONDA CG TIT TITAN
 125 150 160
 Modelo: BIZ C100 HONDA Aplicação: HONDA BIZ C100 125 C125
 Modelo: 100 HONDA BIZ Aplicação: HONDA BIZ C100 125 C125
 Modelo: 100 BIZ BRAVO HONDA Aplicação: HONDA BIZ C100 125 C125
 Modelo: 125 YBR GT YAMAHA Aplicação: YAMAHA FACTOR YBR 125 YBR
 125
 Modelo: 250F TWISTER HONDA Aplicação: HONDA TWISTER CBX 250

Predição e avaliação do Modelo de Regressão Logística

Métricas

In [39]:

```
# tempo de execução para 10.000 classificações
ini=time.time()
#teste=df.sample(10000)['Modelo'].apply(classificaAplicacaoLGR)
teste=df.sample(100)['Modelo'].apply(classificaAplicacaoLGR)
fim=time.time()
```

In [40]:

```
# Realizando a predição
resultadoLGR = clfLGR.predict(X_test)
# Avaliando o modelo
print('Acurácia: {:.2f}'.format(metrics.accuracy_score(y_test, resultadoLGR)))
print('Precisão: {:.2f}'.format(metrics.precision_score(y_test, resultadoLGR, average='micro')))
print('Recall: {:.2f}'.format(metrics.recall_score(y_test, resultadoLGR, average='micro')))
print('F1_Score: {:.2f}'.format(metrics.f1_score(y_test, resultadoLGR, average='micro')))
print("Tempo: " + str(round((fim - ini), 1)) + " segundos.")
```

Acurácia: 0.99
Precisão: 0.99
Recall: 0.99
F1_Score: 0.99
Tempo: 0.0 segundos.

In [41]:

```
# Avaliação completa
print(metrics.classification_report(y_test, resultadolgr))
```

support		precision	recall	f1-score
	SUNDOWN HUNTER	0.96	1.00	0.98
23	BMW F800GS	1.00	1.00	1.00
2	BRAVAX BVX STREET 130	1.00	0.33	0.50
3	DAFRA APACHE	1.00	1.00	1.00
9	DAFRA KANSAS	1.00	1.00	1.00
16	DAFRA NEXT	1.00	1.00	1.00
8	DAFRA RIVA	1.00	1.00	1.00
8	DAFRA SPEED	1.00	1.00	1.00
12	DAFRA SUPER	0.86	1.00	0.92
6	DAFRA ZIG	1.00	1.00	1.00
4	HONDA BIZ C100 125 C125	1.00	1.00	1.00
491	HONDA CB 250 250F	1.00	1.00	1.00
8	HONDA CB 300R 300 CB300	0.99	1.00	1.00
149	HONDA CB 500	1.00	1.00	1.00
1	HONDA CB HORNET 600	1.00	1.00	1.00
6	HONDA CBX 1000	1.00	1.00	1.00
4	HONDA CG 125	1.00	1.00	1.00
33	HONDA CG FAN	1.00	1.00	1.00
887	HONDA CG TIT TITAN 125 150 160	0.99	0.99	0.99
483	HONDA CG TODAY	1.00	0.86	0.92
7	HONDA CRF 230 230F 250 250F	0.97	1.00	0.99
36	HONDA DREAM	1.00	0.95	0.97
19	HONDA NC 700X 700	1.00	1.00	1.00
1	HONDA NX 150 200 250	1.00	0.90	0.95
10	HONDA NX 350 SAHARA	1.00	1.00	1.00
4	HONDA NX 400 FALCON	1.00	1.00	1.00
39	HONDA NXR 150 160 BROZ	1.00	1.00	1.00

628	HONDA POP	1.00	1.00	1.00
247	HONDA STRADA CBX 200	1.00	1.00	1.00
43	HONDA TORNADO XR 250	1.00	1.00	1.00
81	HONDA TWISTER CBX 250	1.00	1.00	1.00
185	HONDA XL 700V TRANSALP	1.00	1.00	1.00
1	HONDA XL XLS 125 XL125 XL125S	1.00	1.00	1.00
60	HONDA XLR	1.00	1.00	1.00
61	HONDA XR	1.00	0.93	0.97
15	HONDA XRE 300	1.00	1.00	1.00
159	KAHENNA TOP	0.83	0.83	0.83
6	KASINSKI COMET GT 150	1.00	1.00	1.00
1	KASINSKI COMET GT 250	1.00	1.00	1.00
1	KASINSKI MIRAGE 150 250	1.00	1.00	1.00
4	KAWASAKI ER-6N ER6N ER-6F ER6F	1.00	1.00	1.00
2	KAWASAKI MAXI	1.00	0.33	0.50
3	KAWASAKI NINJA 250 300	1.00	1.00	1.00
17	KTM EXC 500	0.00	0.00	0.00
1	KTM SX 125	0.00	0.00	0.00
1	KTM SX 150	0.85	1.00	0.92
11	KTM SX 250	0.00	0.00	0.00
3	KTM SX 50	1.00	1.00	1.00
2	KTM XC 525	1.00	1.00	1.00
1	MRX 230R 230	0.00	0.00	0.00
1	MVK MA	0.00	0.00	0.00
3	MVK SPORT	0.00	0.00	0.00
1	SHINERAY JET 50	1.00	0.80	0.89
5	SHINERAY LIBERTY 50	1.00	1.00	1.00
1	SHINERAY PHOENIX 50	0.94	1.00	0.97
16				

4	SHINERAY XY 250	1.00	1.00	1.00
6	SUNDOWN HUNTER	1.00	1.00	1.00
1	SUNDOWN MAX	0.00	0.00	0.00
1	SUNDOWN STX MOTARD	0.00	0.00	0.00
1	SUNDOWN WEB	0.97	1.00	0.99
69	SUZUKI BANDIT GSF 750	1.00	1.00	1.00
1	SUZUKI GS 500	1.00	1.00	1.00
6	SUZUKI GSR	1.00	1.00	1.00
5	SUZUKI INTRUDER	1.00	0.75	0.86
4	SUZUKI KATANA	1.00	0.97	0.98
33	SUZUKI V-STROM DL STROM	0.00	0.00	0.00
1	SUZUKI YES EN 125	0.98	1.00	0.99
60	TRAXX STAR 50	1.00	1.00	1.00
2	TRIUMPH TIGER	1.00	1.00	1.00
1	YAMAHA CRYPTON	1.00	1.00	1.00
72	YAMAHA FACTOR YBR 125 YBR125	1.00	1.00	1.00
385	YAMAHA FAZER FZ6	0.00	0.00	0.00
2	YAMAHA FAZER YS150 150	0.98	1.00	0.99
151	YAMAHA FAZER YS250 250	0.99	1.00	1.00
194	YAMAHA LANDER XTZ 250	1.00	1.00	1.00
124	YAMAHA XJ6	1.00	1.00	1.00
2	YAMAHA XT 225	1.00	1.00	1.00
3	YAMAHA XT 600	1.00	1.00	1.00
3	YAMAHA XT 660R 660	1.00	1.00	1.00
7	YAMAHA XTZ 125	0.98	1.00	0.99
123	YAMAHA XTZ CROSSER 150	1.00	0.99	1.00
123	YAMAHA XTZ TENERE 250	0.86	1.00	0.93
25	YAMAHA YZF R1	1.00	1.00	1.00
9				

	micro avg	0.99	0.99	0.99
5246	macro avg	0.87	0.85	0.86
5246	weighted avg	0.99	0.99	0.99
5246				

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.

```
'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
'precision', 'predicted', average, warn_for)
```

O alerta ao final do relatório é esperado, tendo em vista que existem aplicações raras e que não aparecerem no grupo de teste.

Matriz de Confusão

In [42]:

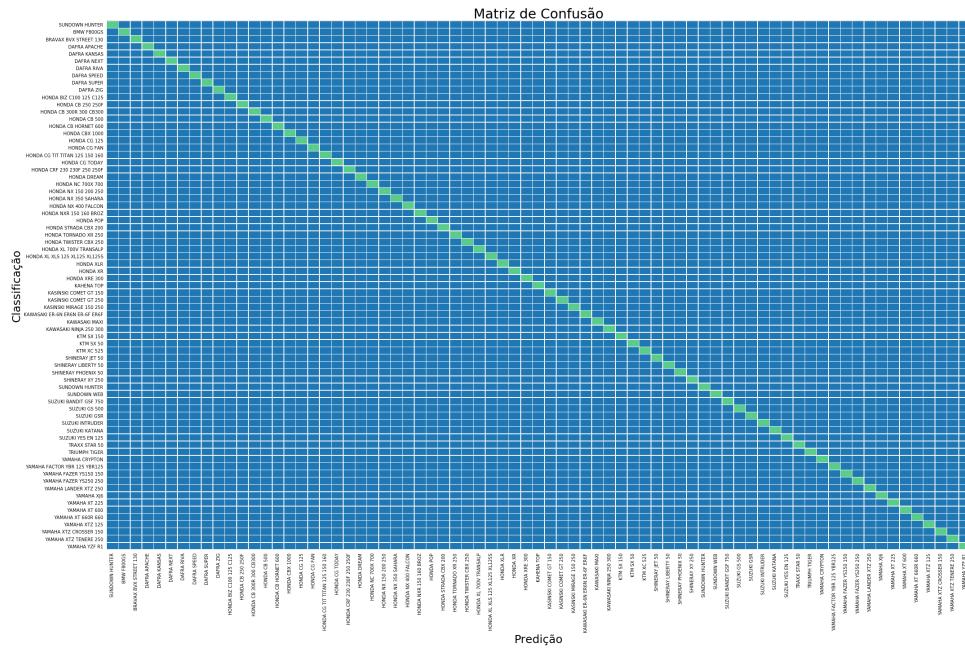
```
# importa os módulos
# gera a matriz de confusão formatada (adaptado por mim)
from prettyPlotConfusionMatrix import pretty_plot_confusion_matrix # pretty
PlotConfusionMatrix.py Local
# código disponível em https://github.com/wcipriano/pretty-print-confusion-
matrix
from sklearn.metrics import confusion_matrix
```

In [43]:

```
aplicacoeslgr = np.unique(resultadolgr) # define os aplicações presentes no
resultado
cmlgr = confusion_matrix(y_test, resultadolgr, aplicacoeslgr) # cria a matriz de confusão
dfmclgr=pd.DataFrame(cmlgr,index=aplicacoeslgr,columns=aplicacoeslgr) # converte a matriz em dataframe
```

In [44]:

```
pretty_plot_confusion_matrix(dfmcngr, annot=True, cmap="tab10", fmt=".0f",
fz=1, lw=0.5, cbar=False,
figsize=[30,20], show_null_values=2, pred_val_
axis='x',insertTot=False)
```

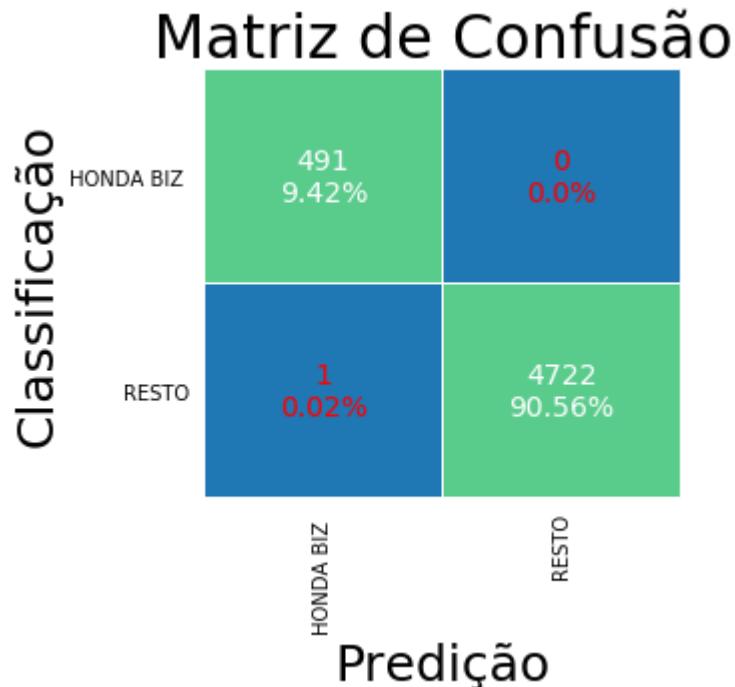


Apesar do grande número de classes, pode-se observar a linha formada pelos acertos na diagonal da imagem, demonstrando o alto índice de acerto do modelo.

Para melhorar a visão, esboçaremos a matriz de confusão para o item 'HONDA BIZ 100 C100 125 C125' versus o resto (OvR), isso nos permitirá ver a classificação como se fosse uma classificação binária OvR do tipo ou é o resto.

In [45]:

```
# gera o plot da matriz de confusão OvR para o modtste do resultadosvc
modteste=dfaplicacoes['APLICACOES'].iloc[276]
plotConfusaoOvR(modteste, aplicacoesslgr, cmngr)
```



Utilização do Modelo

O modelo deverá receber uma descrição do produto conforme entrada do contribuinte na Declaração de Importação e deverá retornar a classificação de aplicação da motocicleta.

Teste com a linha de exemplo

In [46]:

```
print('index:', linha)
descricao = df.iloc[linha]['DESCRICAO DO PRODUTO']
# verifica a descrição do produto
descricao
```

index: 15

Out[46]:

'80372 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE, COROA E PINHÃO PARA MOTOCICLETA RIVA150 DAFRA, MARCA ALLEN.'

In [47]:

```
classificaAplicacaoLGR(criaModelo(descricao))
```

Out[47]:

'DAFRA RIVA'

Escolha do Modelo

Todos os modelos tiveram performance em acertos semelhantes e qualquer um dos escolhidos desempenharia bem o papel de classificar as aplicações.

A escolha tomou por base a performance em tempo de execução, sendo o modelo **Linear SVC** o mais rápido dos três analisados.

Salvando o modelo escolhido (*pickle*)

Para reutilização do modelo posteriormente, vamos salvá-lo em um arquivo serializado pelo módulo pickle e recuperá-lo no momento do uso.

A função *classificaAplicacaoSVC* foi colocada no módulo *funcoesTCC* para importação quando necessário.

In [48]:

```
# salvando com o pickle
# cvt
with open(r'./pickle/cvt.pkl', 'wb') as file:
    pickle.dump(cvt, file)
    file.close
# clfsvc
with open(r'./pickle/clfsvc.pkl', 'wb') as file:
    pickle.dump(clfsvc, file)
    file.close
```

Classificação da Tabela ABIMOTO utilizando o modelo Linear SVC

Pela velocidade, versatilidade, facilidade de uso e, obviamente, a acurácia, optou-se pelo modelo do classificador Linear SVC.

Importando a Tabela ABIMOTO

In [49]:

```
dfABIMOTO = pd.read_excel(r'./bases/dfABIMOTOV13.xlsx')
```

In [50]:

```
dfABIMOTO.sample()
```

Out[50]:

	PARTES E PEÇAS	VMLE	RETENTOR	APLICACAO
11	KIT TRANSMISSÃO 1045 TITAN 150 16D+428HX118+43...	4.3	False	HONDA CG TIT TITAN 125 150 160

Classificando a Tabela ABIMOTO

In [51]:

```
dfABIMOTO['APLICACAO']=dfABIMOTO['PARTES E PEÇAS'].apply(classificaAplicacaoSVC)
```

In [52]:

```
dfABIMOTO.sample()
```

Out[52]:

	PARTES E PEÇAS	VMLE	RETENTOR	APLICACAO
15	KIT TRANSMISSÃO XTZ - 250 LANDER SEM RETENTOR	4.4	False	YAMAHA LANDER XTZ 250

Exportando o DataSet Classificado

Exportando para um arquivo CSV

In [53]:

```
dfABIMOTO.to_csv(r'./bases/dfABIMOT0v13.xlsx', index = False, header = True)
```

Exportando para um arquivo de planilha do Excel

In [54]:

```
dfABIMOTO.to_excel(r'./bases/dfABIMOT0v13.xlsx', index = False, header = True)
```

In [55]:

```
tempotot=time.time()-initot
if tempotot>60:
    print(f'Tempo total de execução: {tempotot/60:.2f} minutos.')
else:
    print(f'Tempo total de execução: {tempotot:.2f} segundos.')
```

Tempo total de execução: 53.52 segundos.

Notebook Jupyter 8_preditãoRisco

Predição do Risco dada uma Importação

O objetivo agora é dada uma importação contendo **kit de transmissão de motocicletas** fazer a sua classificação e em seguida apresentar uma análise de risco visual baseada na estatística existente de outras importações.

É importante salientar que esta análise não substitui a acurada análise a ser realizada por um Auditor Fiscal, tendo em vista que o resultado será uma lista de parâmetros objetivos que indicarão graus de observação para o gerenciamento de risco. Desse modo, a entrada será o registro de importação contendo todos os campos. A função determinará a classificação e com base nos valores declarados de outras importações da mesma classificação, fará a comparação analítica com o valor da importação em análise.

Importando bibliotecas

In [1]:

```
import pandas as pd, time
import pickle
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
```

In [2]:

```
# importa funções criadas no TCC e que precisam ser reutilizadas
from funcoesTCC import *
# Funções: criaModelo, LimpaDescricao, achaPalavraChave, pegaChave, acrescentaMarca, retentorAux, classificaAplicacao
# Variáveis: stopwords, palavrasChave, Marcas, cvt, tfi, clfsvc
# Datasets: dfaplicacoes (Aplicações)
```

In [3]:

```
# Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
```

Código executado em 20/01/2022 às 17:39

Carregamento do dataset

In [4]:

```
# Importa base de dados com os modelos já determinados para um dataframe
df = pd.read_excel(r'./bases/dataframe_modelos_classificado.xlsx')
```

Define dataset para gerar os relatórios

In [5]:

```
dfimp=df.sample(50)
dfimp.sample()
```

Out[5]:

	PAIS DE ORIGEM	DESCRICAÇÃO DO PRODUTO	VALOR UN.PROD.DOLAR	DESCRICAÇÃO	Modelo
12863	CHINA, REPÚBLICA POP	20181/i45 - KIT DE TRANSMISSÃO EM AÇO 1045, MA...	3.56	ybr125	125 yamaha ybr ybr125

Teste do classificador

O classificador treinado e as funções de classificação foram importadas no módulo funcoesTCC

In [6]:

```
descricao = df.iloc[117]['DESCRICAÇÃO DO PRODUTO']
# verifica a descrição do produto
descricao
```

Out[6]:

'KIT DE TRANMISSAO , MARCA RIFFEL, TITANIUM (1045) PARA MOTOCICLETAS XTZ 125, COMPOSTO DE CORRENTE 428 X M138 + COROA 41555 48Z + PINHAO 26571 14Z (CERTIFICADO NR. BR31512030) - ITEM: 71817'

In [7]:

```
classificaAplicacao(descricao)
```

Out[7]:

'YAMAHA XTZ 125'

Importando a Tabela ABIMOTO

In [8]:

```
dfABIMOTO = pd.read_excel(r'./bases/dfABIMOT0v13.xlsx')
```

In [9]:

```
dfABIMOTO.sample()
```

Out[9]:

	PARTES E PEÇAS	VMLE	RETENTOR	APLICACAO
7	KIT TRANSMISSÃO 1045 TITAN 2000/04 14D+428HX11...	4.3	False	HONDA CG TIT TITAN 125 150 160

Função de busca do valor da Aplicação na Tabela ABIMOTO

In [10]:

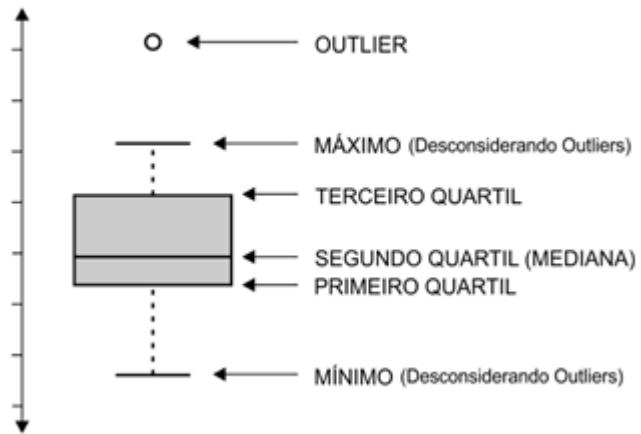
```
def achaValorABIMOTO(aplicacao, retentor):
    # aplicação = string (aplicação da importação atual)
    # retentor = boolean (identificador da presença de retentor na importação atual)
    # retorna o valor na tabela ABIMOTO ou 0 se não existir a aplicação
    dfABtemp=dfABIMOTO[dfABIMOTO['APLICACAO']==aplicacao]
    dfABtemp=dfABtemp[dfABtemp['RETENTOR']==retentor]
    if dfABtemp.shape[0]==0:
        return 0
    return min(dfABtemp['VMLE'])
```

Função de determinação do risco

Uma observação muito importante a ser feita, foi quanto aos valores *outliers*, isto é, valores atípicos, posto que alguns itens eram discrepantes, fosse por ser uma sub ou sobrevaloração do item ou fosse porque alguns importadores insistem em, erroneamente, declarar caixas com vários kits em vez de unidades.

A identificação de *outliers*, distantes dos demais pontos da série, pode ser feita utilizando o intervalo interquartílico (IQR). Entende-se como *outlier*, valores menores que $Q1 - 1,5 \cdot IQR$ ou valores maiores que $Q3 + 1,5 \cdot IQR$.

Estes valores também são representados no boxplot, mas como pontos acima ou abaixo da linha conectada à caixa. Vejamos um desenho de um boxplot com todos estes conceitos:



Apesar da bibliografia indicar o uso de uma vez e meia o intervalo entre o primeiro e o terceiro quartil para definir os *outliers*, optou-se nesse trabalho, por uma questão de segurança, em utilizar o valor de três vezes.

Função de plotagem da Previsão de Risco

In [11]:

```
def plotrisco(P, valor, valorABIMOTO=0):
    # P (Lista) = percentis da base histórica da aplicação importada
    # valor (float) = valor declarado na importação analisada
    # valorABIMOTO (float) = valor de referência na tabela ABIMOTO (0, se não existir)
    # cria o plot
    fig, ax = plt.subplots(figsize=(10, 1.25))

    # define os dados sequenciais, posição e cores dos percentis
    ax.broken_barh([(0,P[0]),(P[0],P[1]), (P[1],P[11]), (P[11],P[3]), (P[3],P[4]), (P[4],P[5]),
                    (P[5],P[6]), (P[6],P[7]), (P[7],P[8]), (P[8],P[9]), (P[9],P[10])],
                   [10, 9],
                   facecolors=('#c10000', 'r', 'r', '#fdff01', '#00fe32', '#02ff00',
                  '#02ff00', '#02ff00', '#00fe32', '#00fe32', '#006dff'))

    # define limites dos eixos x e y
    ax.set_xlim(P[0], P[10])
    ax.set_ylim(10,15)

    # define os marcadores dos eixos
    ax.set_yticks([])
    ax.set_xticks([P[11],P[5],P[9]])
    ax.set_axisbelow(True)

    # define os textos
    # identificação dos percentis
    ax.text(P[1], 15.5, "P10%", fontsize=10, verticalalignment='bottom', horizontalalignment='center', rotation='vertical')
    ax.text(P[1], 15, "|", fontsize=10, verticalalignment='center', horizontalalignment='center')
    ax.text(P[11], 15.5, "P25%", fontsize=10, verticalalignment='bottom', horizontalalignment='center', rotation='vertical')
    ax.text(P[11], 15, "|", fontsize=10, verticalalignment='center', horizontalalignment='center')
    ax.text(P[5], 15.5, "P50%", fontsize=10, verticalalignment='bottom', horizontalalignment='center', rotation='vertical')
    ax.text(P[5], 15, "|", fontsize=10, verticalalignment='center', horizontalalignment='center')
    ax.text(P[12], 15.5, "P75%", fontsize=10, verticalalignment='bottom', horizontalalignment='center', rotation='vertical')
    ax.text(P[12], 15, "|", fontsize=10, verticalalignment='center', horizontalalignment='center')
    ax.text(P[9], 15.5, "P90%", fontsize=10, verticalalignment='bottom', horizontalalignment='center', rotation='vertical')
    ax.text(P[9], 15, "|", fontsize=10, verticalalignment='center', horizontalalignment='center')
    ax.text(P[10], 15.5, "P100%", fontsize=10, verticalalignment='bottom', horizontalalignment='center', rotation='vertical')
    ax.text(P[10], 15, "|", fontsize=10, verticalalignment='center', horizontalalignment='center')
```

```

# o valor (P[10]-P[0])/40 relativiza a escala na hora de mover algo no
gráfico.
undrelativa = (P[10]-P[0])/40
# identificação de mínimo, mediana e máximo
ax.text(P[0]-1*undrelativa, 12.5, f"Min({P[0]})", fontsize=12, vertical
alignment='center', horizontalalignment='center', rotation='vertical')
ax.text((P[5]), 8, "Mediana", fontsize=12, verticalalignment='center',
horizontalalignment='center')
ax.text(P[10]+1*undrelativa, 12.5, f"Máx({P[10]})", fontsize=12, vertic
alalignment='center', horizontalalignment='center', rotation='vertical')

# define alertas
alerta=''
# alerta quando valor fora dos limites máximo e mínimo
if valor<P[0] or valor>P[10]:
    alerta+=f"ATENÇÃO: Valor declarado de USD{valor:.2f} fora dos limit
es do modelo (outlier)."
    alerta+='\n'
if valorABIMOTO>valor:
    alerta+=f"ATENÇÃO: Valor declarado de USD{valor:.2f} menor que refe
rência mínima ABIMOTO: USD{valorABIMOTO:.2f}."
    ax.text((P[10]+P[0])/2, 6, alerta, fontsize=13, color='r', verticalalignme
nt='center', horizontalalignment='center')

# define o título
fig.suptitle('Previsão de Risco', fontsize=13, y=1.5, horizontalalignme
nt='center')

# define cores do ponteiro indicador
cores=['#a20000','#ff6900', '#007d00', '#00007d'] # vermelho, amarelo,
verde, azul
# condições para determinar a cor do ponteiro
if valor>=P[9]: cor=cores[3]
elif valor>=P[3]: cor=cores[2]
elif valor>=P[11]: cor=cores[1]
else: cor=cores[0]

# plota o ponteiro e o valor
ax.scatter(x=valor, y=10.9, marker='v', c=cor, s=400)
ax.text(valor, 12.2, valor, color='black', fontsize=14, verticalalignme
nt='center', horizontalalignment='center')
ax.vlines(x=P[5], ymin=10, ymax=15)

# se existir valor na tabela ABIMOTO coloca o indicador e valor
if valorABIMOTO>0:
    ax.scatter(x=valorABIMOTO, y=10.7, marker='v', c='black', s=200)
    if valorABIMOTO>valor:
        ax.text(valorABIMOTO+1*undrelativa, 10.6, f'{valorABIMOTO:.2f}
(ABIMOTO)', color='black', fontsize=12, verticalalignment='center', horizon
talalignment='left')
    else:
        ax.text(valorABIMOTO-1*undrelativa, 10.6, f'(ABIMOTO){valorABIM
OTO:.2f}', color='black', fontsize=12, verticalalignment='center', horizon
talalignment='right')

# mostra o gráfico
plt.show()

```

Função que calcula o risco de valoração

In [12]:

```
def riscoValor(dfimp):
    # dfimp será uma série ou dataframe pandas contendo as importações a analisar
    # caso seja um dataframe e tenha mais de uma linha, será iterada sobre todas as linhas.
    if type(dfimp)==type(pd.Series()): tipo='s' # se for tipo series => s
    if type(dfimp)==type(pd.DataFrame()): tipo='d' # se for tipo dataframe => d
    if not(tipo=='s' or tipo=='d'): # se o tipo não for séries ou dataframe => erro
        raise TypeError('É preciso entrar com um dataframe ou uma Series do pandas.')
    if tipo=='d': # se for do tipo dataframe
        for i, linha in dfimp.iterrows(): # executa a função novamente para cada linha (series)
            riscoValor(linha) # reexecuta para cada linha se fornecido um dataframe (recursividade)
    if tipo=='s':
        print('\x1b[1;31m'+ 'RELATÓRIO DE RISCO' + '\x1b[0m')
        print('\x1b[1;31m'+ '\nDados da Declaração de Importação' + '\x1b[0m')
        descricao=dfimp['DESCRICAO DO PRODUTO']
        origem=dfimp['PAIS DE ORIGEM']
        retentor=retentorAux(dfimp['DESCRICAO DO PRODUTO'])
        retentortexto="com retentor" if retentor else "sem retentor"
        aplicacao=classificaAplicacao(descricao)
        valor=round(dfimp['VALOR UN.PROD.DOLAR'],2)
        valorABIMOTO=achaValorABIMOTO(aplicacao, retentor)
        print(f'Descrição: {descricao}\n'+
              f'Origem: {origem}\n'+
              f'Retentor: {retentortexto}\n'+
              f'Aplicação: {aplicacao}\n'+
              f'Valor DI: USD {valor:.2f}')
        if valorABIMOTO>0: print(f'Tabela de Referência ABIMOTO\nValor: USD {valorABIMOTO:.2f}\n')
        # filtra o df somente para os registros da aplicação classificada
        dfrisco=df[df['APLICACAO']==aplicacao] # filtra para a aplicação
        dfrisco=dfrisco[dfrisco['RETENTOR']==retentor] # filtra para a existência de retentor
        # remove os valores discrepantes dos dados
        # filtra somente valores de Q25-discrepância*(Q75-Q25) a Q75+*discrepância*(Q75-Q25)
        discrepancia=3 # valor padrão=1.5
        P25=np.percentile(dfrisco['VALOR UN.PROD.DOLAR'],25)
        P75=np.percentile(dfrisco['VALOR UN.PROD.DOLAR'],75)
        min=P25-discrepacia*(P75-P25)
        max=P75+discrepacia*(P75-P25)
        filtro=[min<=dfrisco['VALOR UN.PROD.DOLAR']] and [dfrisco['VALOR UN.PROD.DOLAR']<=max]
        dfrisco=dfrisco[filtro[0]]
        # recálculo dos quartis 25 e 75 para df sem outliers
        P25=np.percentile(dfrisco['VALOR UN.PROD.DOLAR'],25)
        P75=np.percentile(dfrisco['VALOR UN.PROD.DOLAR'],75)
        # geração da lista com os quartis
```

```

P=[round(np.percentile(dfrisco['VALOR UN.PROD.DOLAR'],x),2) for x in
range(0,110,10)]+[round(P25,2),round(P75,2)]
estatisticas=dfrisco.describe()['VALOR UN.PROD.DOLAR']
estatisticas.rename(index={'count':'qtd de registros','mean':'média simples','std':'desvio padrão','min':'valor mínimo','25%':'percentil 5%','50%':'percentil 50%','75%':'percentil 75%','max':'valor máximo'},inplace=True)
print('\x1b[1;31m'+f'Estatísticas:\n'+ '\x1b[0m'+f'{estatisticas.to_
string()}\n')
# fazer um plot indicando a posição do valor em uma barra variando
em vermelho-amarelo-verde-azul
quartil={}
for x in range(0,10): quartil[10+x*10] = P[x]
quartil[25]=P25
quartil[75]=P75
perc=[f'{chave}={quartil[chave]:.2f}' for chave in sorted(quartil.keys())]
print('\x1b[1;31m'+f'Percentis:\n'+ '\x1b[0m'+f'{str(perc[:6])[1:-1]}\n {str(perc[6:])[1:-1]}')
print('\x1b[1;31m'+f'\nHistograma:' + '\x1b[0m')
plt.figure(figsize=(10,5),frameon=False)
plt.hist(dfrisco['VALOR UN.PROD.DOLAR'],10,rwidth=0.9,range=(P[0],P[10]))
plt.show()
print('\x1b[1;31m'+f'Previsão de Risco:' + '\x1b[0m')
plotrisco(P,valor,valorABIMOTO)
print('\n\n\n')

```

In [13]:

```
riscoValor(df.iloc[2103])
```

RELATÓRIO DE RISCO

Dados da Declaração de Importação

Descrição: 22317 - 91131 - KIT DE TRANSMISSÃO PARA MOTOCICLETA
- MODELO C 100 BIZ (13-15) - CONTENDO COROA 34Z - PINHAO 14Z -
C/CORRENTE 428H X 108L - TITANIUM (1045)
Origem: CHINA, REPUBLICA POP
Retentor: sem retentor
Aplicação: HONDA BIZ C100 125 C125
Valor DI: USD 4.09
Tabela de Referência ABIMOTO
Valor: USD 3.90

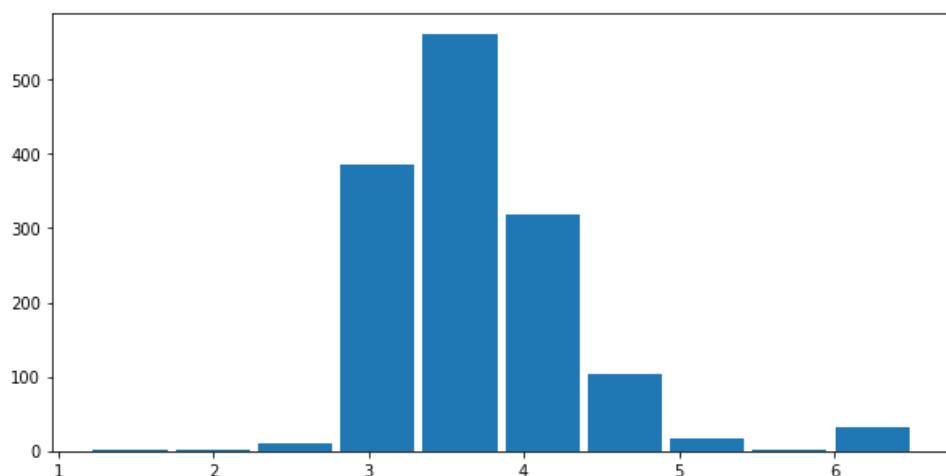
Estatísticas:

qtd de registros:	1434.000000
média simples:	3.742230
desvio padrão:	0.618322
valor mínimo:	1.202000
percentil 25%:	3.290000
percentil 50%:	3.713600
percentil 75%:	4.090000
valor máximo:	6.511000

Percentis:

'10=1.20', '20=3.06', '25=3.29', '30=3.23', '40=3.36', '50=3.55'
'60=3.71', '70=3.79', '75=4.09', '80=3.94', '90=4.13', '100=4.48'

Histograma:



Previsão de Risco:

