

# ***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 tamanho 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 cont  
rário.  
filtro=[]  
for i, aplicacao in enumerate(df['APLICACAOFIM']):  
    if aplicacao[0]=='[':  
        filtro.append(True)  
    else:  
        filtro.append(False)
```

In [7]:

```
df[filtro].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	KAHENA TOP	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	KAHENA TOP	HONDA TWISTER CBX 250	HONDA CB 250 250F

	DESCRICAO	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	KAHENA 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	KAHENA 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	xl7 700 transalp	700 honda transalp	HONDA XL 700V TRANSALP	KAWASAKI NINJA 700	HONDA NC 700X 700
6170	xl7 700 transalp	700 honda transalp	HONDA XL 700V TRANSALP	KAWASAKI NINJA 700	HONDA NC 700X 700

	DESCRICAO	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	KAHENA TOP	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	xl v 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

	DESCRICAO	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	KAHENA TOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
12707	cb twister 250 top	250 cb honda top twister	KAHENA 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



	DESCRICAO	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	KAHENA TOP	HONDA TWISTER CBX 250	HONDA CB 250 250F
15977	cb twister 250 top	250 cb honda top twister	KAHENA 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['DESCRICAO 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(aplicacaoofim):
    aplicacaoofim=" ".join(aplicacaoofim)
    aplicacaoofim=aplicacaoofim.replace("'", '')
    aplicacaoofim=aplicacaoofim.replace("[", '')
    aplicacaoofim=aplicacaoofim.replace("]", '')
    aplicacaoofim=aplicacaoofim.split(",")
    return aplicacaoofim
```

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

Obsrva-se que são pouquíssimos casos a analisar.

In [12]:

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

Out[12]:

	Modelo	APLICACAOFIM
2	700 honda transalp	[HONDA XL 700V TRANSALP, 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 cont
rário.
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 dicionário, 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 pa
ra as verificações até o índice i
        break
    if row['APLICACAOFIM'][0]=='[':
        # guarda a lista das opções na variável
        aplictemp=row['APLICACAOFIM'].replace("[", "").replace(", ", ";").rep
lace(", ", ";").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 op
ções 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 p
ickle
```

```

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 cont
rário.
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('APLICACAO SVC', 1)
df = df.drop('APLICACAO MNB', 1)
df = df.drop('APLICACAO FIM', 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.