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 predições 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.localt
ime(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='s
trict',
        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=Non
e,
        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='s
trict',
        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=Non
e,
        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_aplicacoe
s)[-500:]) # amostra
```

```
{'acellera': 85, 'acx': 87, '250f': 31, '250': 30, 'frontlande
r': 327, '500': 47, '800': 72, 'efi': 274, 'hotzoo': 408, 'spo
rt': 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': 8
9, '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 (...) 783, 'tzr': 784, 'v-max': 787, '1680': 22, 'v-star': 7
90, 'virago': 807, '535': 51, '450f': 43, 'xf50x': 850, 'xj6':
851, 'xjr': 852, 'xr180': 862, 'xs1100': 867, 'xs': 866, 'xs40
0': 868, 'xs500': 869, 'xs650': 870, 'xs750': 871, 'xs850': 87
2, 'xt': 873, '225': 26, '660r': 59, '660': 58, 'tenere': 738,
'1200z': 12, '660z': 60, 'crosser': 210, 'xz': 877, 'yfm': 88
3, '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:]}') #
amostra
```

```
  (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	0
1	0	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	0
3	0	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	0

5 rows × 905 columns

In [18]:

```
# Totalização do número de termos de cada vocábulo
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 aparacem 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'].ilo
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 apli
cação 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 funão 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.
```