Notebook Jupyter 7_treinamentoClassificador

Classificação da Aplicação por aprendizado de máquina

Importando bibliotecas

In [1]:
```
import pandas as pd, numpy as np, 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/Treste
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, acrescentaMarc
a, retentorAux, classificaAplicacaoSVC
# Variávis: stopwords, palavrasChave, Marcas, cvt, tfi, clfsvc
# Datasets: dfaplicacoes (Aplicações)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\47929790304\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

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(tim
e.time()))}')
```

```
Código executado em 25/01/2022 às 13:31
```

Carregando 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')
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	Modelo	APLICACAO	RETEN
2662	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO , MARCA RIFFEL, TITANIUM (1...	4.56570	cg 150 titan 160 cargo fan start	150 160 cargo cg fan honda titan	HONDA CG FAN	F
9556	CHINA, REPUBLICA POP	TITAN 00 - KIT TRANSMISSAO PARA MOTOCICLETA, C...	3.50671	titan	honda titan	HONDA CG TIT TITAN 125 150 160	F
13925	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO PARA MOTOCICLETAS MODELO: W...	4.68200	web	sundown web	SUNDOWN WEB	F
16216	CHINA, REPUBLICA POP	KIT DE TRANSMISSAO, MARCA RIFFEL, TITANIUM (10...	6.55500	ys 250 fazer fz	250 fazer yamaha	YAMAHA FAZER YS250 250	F
8418	CHINA, REPUBLICA POP	item 22;Partes e peças para Motocicletas,Desta...	5.94660	katana 125 en yes cargo gs	125 cargo en gs katana suzuki yes	SUZUKI YES EN 125	F

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 PINHÃO PARA MOTOCICLETA R
IVA150 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

Análise do uso do TF-IDF

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”.
Após diversos testes observou-se que o uso do TF-IDF trazia várias classificações inadequadas ao conjunto de dados.
Decidiu-se, então, pela não utilização do ponderador TF-IDF.

In [9]:
```
# Criação da função CountVectorizer
cvt = CountVectorizer(strip_accents='ascii', lowercase=True)
X_cvt = cvt.fit_transform(df['Modelo'])
```

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 saida será as aplicações
saida = df['APLICACAO']
# Separando 30% dos dados para teste
X_train, X_test, y_train, y_test = train_test_split(entrada, saida, 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()
```

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 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 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,averag
e='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='micr
o')))
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))
```

	precision	recall	f1-score	support
BMW F800GS	1.00	1.00	1.00	1
BRAVAX BVX STREET 130	1.00	0.67	0.80	3
DAFRA APACHE	1.00	1.00	1.00	12
DAFRA KANSAS	1.00	1.00	1.00	17
DAFRA NEXT	1.00	1.00	1.00	9
DAFRA RIVA	1.00	1.00	1.00	18
DAFRA SPEED	1.00	1.00	1.00	11
DAFRA SUPER	0.86	1.00	0.92	6
DAFRA ZIG	1.00	1.00	1.00	2
HONDA BIZ C100 125 C125	1.00	1.00	1.00	470
HONDA CB 250 250F	1.00	1.00	1.00	13
HONDA CB 300R 300 CB300	1.00	1.00	1.00	132
HONDA CB 500	1.00	1.00	1.00	4
HONDA CB HORNET 600	1.00	1.00	1.00	3
HONDA CBR 600	0.00	0.00	0.00	1
HONDA CBX 1000	1.00	1.00	1.00	4
HONDA CG 125	1.00	1.00	1.00	26
HONDA CG FAN	1.00	1.00	1.00	851
HONDA CG TIT TITAN 125 150 160	0.99	0.99	0.99	493
HONDA CG TODAY	1.00	0.75	0.86	4
HONDA CRF 230 230F 250 250F	1.00	1.00	1.00	33
HONDA DREAM	1.00	0.87	0.93	15
HONDA NC 700X 700	1.00	1.00	1.00	2
HONDA NX 150 200 250	1.00	1.00	1.00	11
HONDA NX 350 SAHARA	1.00	1.00	1.00	6
HONDA NX 400 FALCON	1.00	1.00	1.00	57
HONDA NXR 150 160 BROZ	1.00	1.00	1.00	665
HONDA POP	1.00	1.00	1.00	229
HONDA STRADA CBX 200	1.00	1.00	1.00	47
HONDA TORNADO XR 250	1.00	1.00	1.00	88
HONDA TWISTER CBX 250	1.00	1.00	1.00	187
HONDA XL 700V TRANSALP	1.00	1.00	1.00	1
HONDA XL XLS 125 XL125 XL125S	1.00	1.00	1.00	46
HONDA XLR	1.00	1.00	1.00	70
HONDA XR	1.00	1.00	1.00	22
HONDA XRE 300	1.00	1.00	1.00	152
KAHENA TOP	1.00	1.00	1.00	5
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	5
KASINSKI WIN	0.00	0.00	0.00	1
KAWASAKI MAXI	1.00	0.25	0.40	4
KAWASAKI NINJA 250 300	1.00	1.00	1.00	19
KAWASAKI NINJA 600	0.50	1.00	0.67	1
KAWASAKI Z800	0.00	0.00	0.00	1
KTM SX 125	1.00	1.00	1.00	3
KTM SX 150	1.00	1.00	1.00	11
KTM SX 250	1.00	1.00	1.00	1
KTM SX 50	1.00	1.00	1.00	4
KTM XC 525	1.00	1.00	1.00	3
MVK MA	0.50	1.00	0.67	1
MVK SPORT	0.00	0.00	0.00	1
SHINERAY JET 50	0.75	0.75	0.75	4
SHINERAY LIBERTY 50	1.00	1.00	1.00	1
SHINERAY PHOENIX 50	1.00	0.90	0.95	10
SHINERAY XY 250	1.00	1.00	1.00	2
SUNDOWN HUNTER	1.00	1.00	1.00	22
SUNDOWN MAX	1.00	1.00	1.00	3
SUNDOWN STX MOTARD	1.00	1.00	1.00	1
SUNDOWN WEB	0.99	1.00	0.99	81
SUZUKI BANDIT GSF 750	1.00	1.00	1.00	1
SUZUKI GS 500	1.00	1.00	1.00	8
SUZUKI GSR	1.00	1.00	1.00	11
SUZUKI INTRUDER	1.00	0.83	0.91	6
SUZUKI KATANA	0.97	0.97	0.97	29
SUZUKI YES EN 125	0.99	0.99	0.99	78
TRAXX STAR 50	1.00	1.00	1.00	3
TRIUMPH TIGER	0.67	1.00	0.80	2

```
                YAMAHA CRYPTON       1.00      1.00      1.00        84
   YAMAHA FACTOR YBR 125 YBR125      1.00      1.00      1.00       362
         YAMAHA FAZER YS150 150      0.99      1.00      1.00       142
         YAMAHA FAZER YS250 250      0.99      1.00      1.00       182
          YAMAHA LANDER XTZ 250      1.00      1.00      1.00       138
                     YAMAHA XJ6      1.00      1.00      1.00         4
                  YAMAHA XT 600      1.00      1.00      1.00         3
             YAMAHA XT 660R 660      1.00      1.00      1.00         7
                 YAMAHA XTZ 125      1.00      1.00      1.00       126
         YAMAHA XTZ CROSSER 150      1.00      1.00      1.00       120
          YAMAHA XTZ TENERE 250      1.00      1.00      1.00        36
                     YAMAHA YFS      1.00      1.00      1.00         2
                  YAMAHA YZF R6      1.00      1.00      1.00         5

                       accuracy                          1.00      5246
                      macro avg      0.93      0.93      0.92      5246
                   weighted avg      1.00      1.00      1.00      5246

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:131
8: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:131
8: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:131
8: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

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

Matriz de Confusão

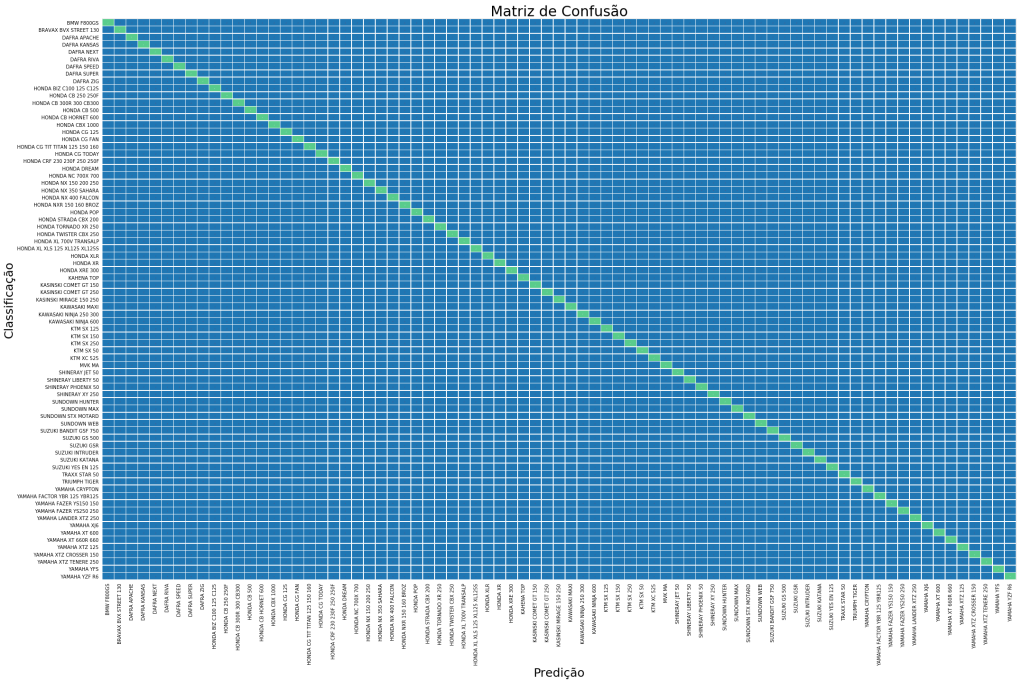
In [18]:

```
# importa os módulos
# gera a matriz de confusão formatada (adaptado por mim)
from prettyPlotConfusionMatrix import pretty_plot_confusion_matrix # prettyPlotConf
usionMatrix.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 os aplicações presentes no resulta
do
cmsvc = confusion_matrix(y_test, resultadosvc, labels=aplicacoessvc) # cria a matri
z 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)
```

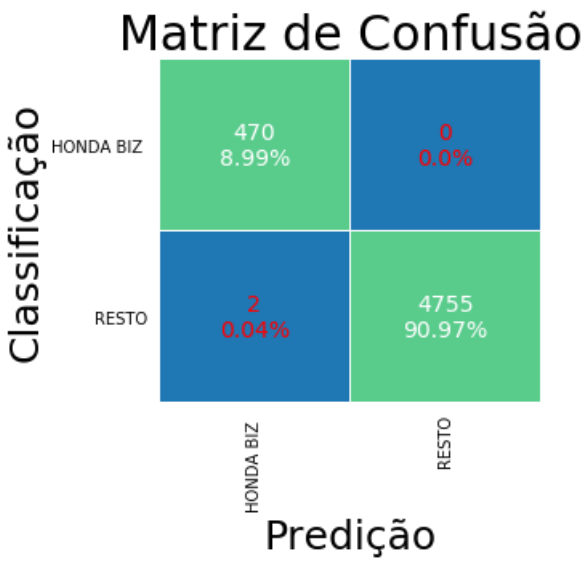


Apesar do grande número de classes, pode-se observar a linha formada pelos acertos na diagonal da imagem, demonstrando o alto índice de acerto do modelo.

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 cara
cteres
    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 PINHÃO PARA MOTOCICLETA R
IVA150 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()
```

Funcão de classificacão

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 YBR125
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,averag
e='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='micr
o')))
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))
```

	precision	recall	f1-score	support
BMW F800GS	0.50	1.00	0.67	1
BRAVAX BVX STREET 130	0.00	0.00	0.00	3
DAFRA APACHE	1.00	1.00	1.00	12
DAFRA KANSAS	1.00	1.00	1.00	17
DAFRA NEXT	1.00	1.00	1.00	9
DAFRA RIVA	1.00	1.00	1.00	18
DAFRA SPEED	1.00	1.00	1.00	11
DAFRA SUPER	1.00	1.00	1.00	6
DAFRA ZIG	1.00	1.00	1.00	2
HONDA BIZ C100 125 C125	0.99	1.00	0.99	470
HONDA CB 250 250F	1.00	0.31	0.47	13
HONDA CB 300R 300 CB300	0.97	1.00	0.99	132
HONDA CB 500	0.00	0.00	0.00	4
HONDA CB HORNET 600	0.67	0.67	0.67	3
HONDA CBR 600	0.00	0.00	0.00	1
HONDA CBX 1000	0.00	0.00	0.00	4
HONDA CG 125	1.00	0.04	0.07	26
HONDA CG FAN	0.89	1.00	0.94	851
HONDA CG TIT TITAN 125 150 160	0.98	0.83	0.90	493
HONDA CG TODAY	0.00	0.00	0.00	4
HONDA CRF 230 230F 250 250F	1.00	1.00	1.00	33
HONDA DREAM	1.00	0.73	0.85	15
HONDA NC 700X 700	0.00	0.00	0.00	2
HONDA NX 150 200 250	1.00	1.00	1.00	11
HONDA NX 350 SAHARA	1.00	0.67	0.80	6
HONDA NX 400 FALCON	0.97	1.00	0.98	57
HONDA NXR 150 160 BROZ	0.94	1.00	0.97	665
HONDA POP	0.97	1.00	0.98	229
HONDA STRADA CBX 200	0.80	1.00	0.89	47
HONDA TORNADO XR 250	1.00	1.00	1.00	88
HONDA TWISTER CBX 250	0.94	1.00	0.97	187
HONDA XL 700V TRANSALP	0.00	0.00	0.00	1
HONDA XL XLS 125 XL125 XL125S	1.00	1.00	1.00	46
HONDA XLR	1.00	1.00	1.00	70
HONDA XR	0.00	0.00	0.00	22
HONDA XRE 300	1.00	0.89	0.94	152
KAHENA TOP	0.00	0.00	0.00	5
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	5
KASINSKI WIN	0.00	0.00	0.00	1
KAWASAKI MAXI	0.00	0.00	0.00	4
KAWASAKI NINJA 250 300	1.00	1.00	1.00	19
KAWASAKI NINJA 600	0.00	0.00	0.00	1
KAWASAKI Z800	0.00	0.00	0.00	1
KTM SX 125	0.00	0.00	0.00	3
KTM SX 150	0.00	0.00	0.00	11
KTM SX 250	0.00	0.00	0.00	1
KTM SX 50	0.00	0.00	0.00	4
KTM XC 525	1.00	0.33	0.50	3
MVK MA	0.00	0.00	0.00	1
MVK SPORT	0.00	0.00	0.00	1
SHINERAY JET 50	1.00	0.75	0.86	4
SHINERAY LIBERTY 50	0.00	0.00	0.00	1
SHINERAY PHOENIX 50	0.83	1.00	0.91	10
SHINERAY XY 250	0.00	0.00	0.00	2
SUNDOWN HUNTER	0.92	1.00	0.96	22
SUNDOWN MAX	0.00	0.00	0.00	3
SUNDOWN STX MOTARD	0.00	0.00	0.00	1
SUNDOWN WEB	0.89	1.00	0.94	81
SUZUKI BANDIT GSF 750	1.00	1.00	1.00	1
SUZUKI GS 500	1.00	1.00	1.00	8
SUZUKI GSR	1.00	1.00	1.00	11
SUZUKI INTRUDER	0.00	0.00	0.00	6
SUZUKI KATANA	1.00	0.69	0.82	29
SUZUKI YES EN 125	0.84	1.00	0.91	78
TRAXX STAR 50	1.00	1.00	1.00	3
TRIUMPH TIGER	1.00	1.00	1.00	2

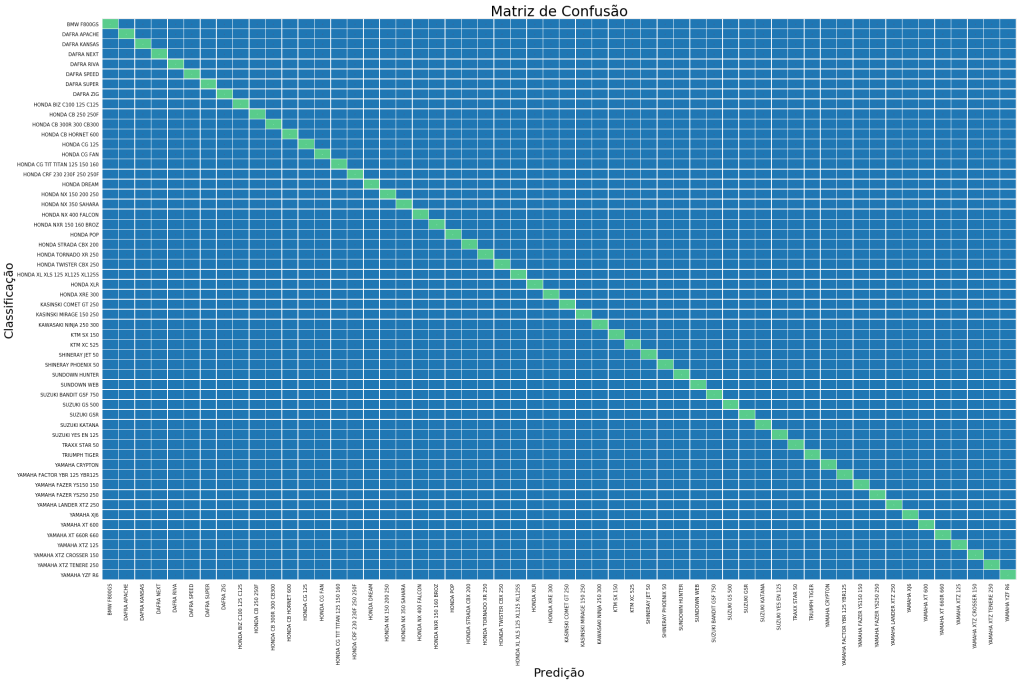
```
                  YAMAHA CRYPTON       1.00      1.00      1.00        84
  YAMAHA FACTOR YBR 125 YBR125       0.99      1.00      1.00       362
        YAMAHA FAZER YS150 150       0.99      1.00      1.00       142
        YAMAHA FAZER YS250 250       0.97      1.00      0.98       182
         YAMAHA LANDER XTZ 250       0.82      1.00      0.90       138
                    YAMAHA XJ6       1.00      1.00      1.00         4
                 YAMAHA XT 600       0.75      1.00      0.86         3
            YAMAHA XT 660R 660       1.00      1.00      1.00         7
                YAMAHA XTZ 125       0.99      1.00      1.00       126
        YAMAHA XTZ CROSSER 150       1.00      1.00      1.00       120
         YAMAHA XTZ TENERE 250       1.00      0.14      0.24        36
                    YAMAHA YFS       0.00      0.00      0.00         2
                 YAMAHA YZF R6       1.00      1.00      1.00         5

                      accuracy                           0.95      5246
                     macro avg       0.65      0.62      0.62      5246
                  weighted avg       0.94      0.95      0.93      5246

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:131
8: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:131
8: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:131
8: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

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

Matriz de Confusão

In [31]:
```
aplicacoesmnb = np.unique(resultadomnb) # define os aplicações presentes no resulta
do
cmmnb = confusion_matrix(y_test, resultadomnb, labels=aplicacoesmnb) # cria a matri
z de confusão
dfmcmnb=pd.DataFrame(cmmnb,index=aplicacoesmnb,columns=aplicacoesmnb) # converte a
 matriz em dataframe
```

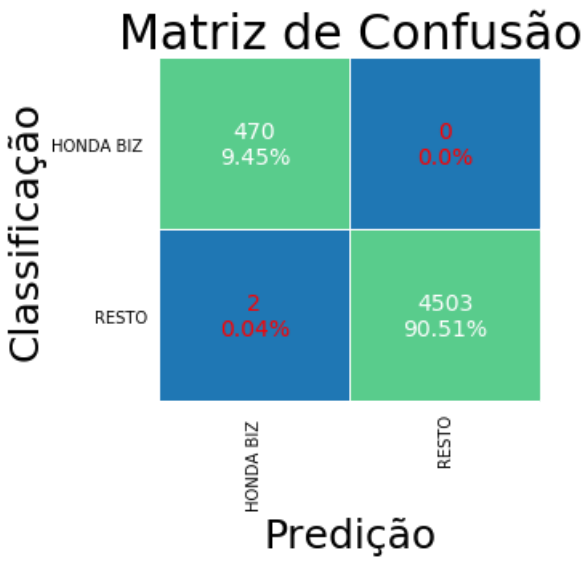
In [32]:
```
pretty_plot_confusion_matrix(dfmcmnb, annot=True, cmap="tab10", fmt='.0f', fz=1, lw
=0.5, cbar=False,
                             figsize=[30,20], show_null_values=2, pred_val_axis='x'
,insertTot=False)
```



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 R
IVA150 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(multi_class='multinomial')
```

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 YBR125
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,averag
e='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='micr
o')))
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))
```

	precision	recall	f1-score	support
BMW F800GS	1.00	1.00	1.00	1
BRAVAX BVX STREET 130	1.00	0.67	0.80	3
DAFRA APACHE	1.00	1.00	1.00	12
DAFRA KANSAS	1.00	1.00	1.00	17
DAFRA NEXT	1.00	1.00	1.00	9
DAFRA RIVA	1.00	1.00	1.00	18
DAFRA SPEED	1.00	1.00	1.00	11
DAFRA SUPER	0.75	1.00	0.86	6
DAFRA ZIG	1.00	1.00	1.00	2
HONDA BIZ C100 125 C125	1.00	1.00	1.00	470
HONDA CB 250 250F	1.00	1.00	1.00	13
HONDA CB 300R 300 CB300	1.00	1.00	1.00	132
HONDA CB 500	1.00	1.00	1.00	4
HONDA CB HORNET 600	0.75	1.00	0.86	3
HONDA CBR 600	0.00	0.00	0.00	1
HONDA CBX 1000	1.00	1.00	1.00	4
HONDA CG 125	1.00	1.00	1.00	26
HONDA CG FAN	1.00	1.00	1.00	851
HONDA CG TIT TITAN 125 150 160	0.99	0.99	0.99	493
HONDA CG TODAY	1.00	0.75	0.86	4
HONDA CRF 230 230F 250 250F	1.00	1.00	1.00	33
HONDA DREAM	1.00	0.87	0.93	15
HONDA NC 700X 700	1.00	0.50	0.67	2
HONDA NX 150 200 250	0.85	1.00	0.92	11
HONDA NX 350 SAHARA	1.00	0.67	0.80	6
HONDA NX 400 FALCON	1.00	1.00	1.00	57
HONDA NXR 150 160 BROZ	1.00	1.00	1.00	665
HONDA POP	1.00	1.00	1.00	229
HONDA STRADA CBX 200	1.00	1.00	1.00	47
HONDA TORNADO XR 250	1.00	1.00	1.00	88
HONDA TWISTER CBX 250	1.00	1.00	1.00	187
HONDA XL 700V TRANSALP	1.00	1.00	1.00	1
HONDA XL XLS 125 XL125 XL125S	0.90	1.00	0.95	46
HONDA XLR	1.00	1.00	1.00	70
HONDA XR	1.00	1.00	1.00	22
HONDA XRE 300	1.00	1.00	1.00	152
KAHENA TOP	1.00	0.60	0.75	5
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	5
KASINSKI WIN	0.00	0.00	0.00	1
KAWASAKI MAXI	1.00	0.25	0.40	4
KAWASAKI NINJA 250 300	1.00	1.00	1.00	19
KAWASAKI NINJA 600	1.00	1.00	1.00	1
KAWASAKI Z800	0.00	0.00	0.00	1
KTM SX 125	0.00	0.00	0.00	3
KTM SX 150	1.00	1.00	1.00	11
KTM SX 250	0.00	0.00	0.00	1
KTM SX 50	0.00	0.00	0.00	4
KTM XC 525	1.00	1.00	1.00	3
MVK MA	0.00	0.00	0.00	1
MVK SPORT	0.00	0.00	0.00	1
SHINERAY JET 50	1.00	0.75	0.86	4
SHINERAY LIBERTY 50	1.00	1.00	1.00	1
SHINERAY PHOENIX 50	1.00	1.00	1.00	10
SHINERAY XY 250	1.00	1.00	1.00	2
SUNDOWN HUNTER	1.00	1.00	1.00	22
SUNDOWN MAX	0.00	0.00	0.00	3
SUNDOWN STX MOTARD	0.00	0.00	0.00	1
SUNDOWN WEB	0.96	1.00	0.98	81
SUZUKI BANDIT GSF 750	1.00	1.00	1.00	1
SUZUKI GS 500	1.00	1.00	1.00	8
SUZUKI GSR	1.00	1.00	1.00	11
SUZUKI INTRUDER	1.00	0.83	0.91	6
SUZUKI KATANA	0.97	0.97	0.97	29
SUZUKI YES EN 125	0.99	0.99	0.99	78
TRAXX STAR 50	0.43	1.00	0.60	3
TRIUMPH TIGER	0.67	1.00	0.80	2

```
                 YAMAHA CRYPTON       1.00      1.00      1.00        84
         YAMAHA FACTOR YBR 125 YBR125       0.99      1.00      1.00       362
             YAMAHA FAZER YS150 150       0.99      1.00      1.00       142
             YAMAHA FAZER YS250 250       0.98      1.00      0.99       182
              YAMAHA LANDER XTZ 250       1.00      1.00      1.00       138
                         YAMAHA XJ6       1.00      1.00      1.00         4
                      YAMAHA XT 600       1.00      1.00      1.00         3
                 YAMAHA XT 660R 660       1.00      1.00      1.00         7
                     YAMAHA XTZ 125       1.00      1.00      1.00       126
             YAMAHA XTZ CROSSER 150       1.00      0.99      1.00       120
              YAMAHA XTZ TENERE 250       0.97      1.00      0.99        36
                         YAMAHA YFS       0.00      0.00      0.00         2
                      YAMAHA YZF R6       1.00      1.00      1.00         5

                           accuracy                           0.99      5246
                          macro avg       0.84      0.82      0.83      5246
                       weighted avg       0.99      0.99      0.99      5246
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:131
8: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:131
8: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:131
8: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

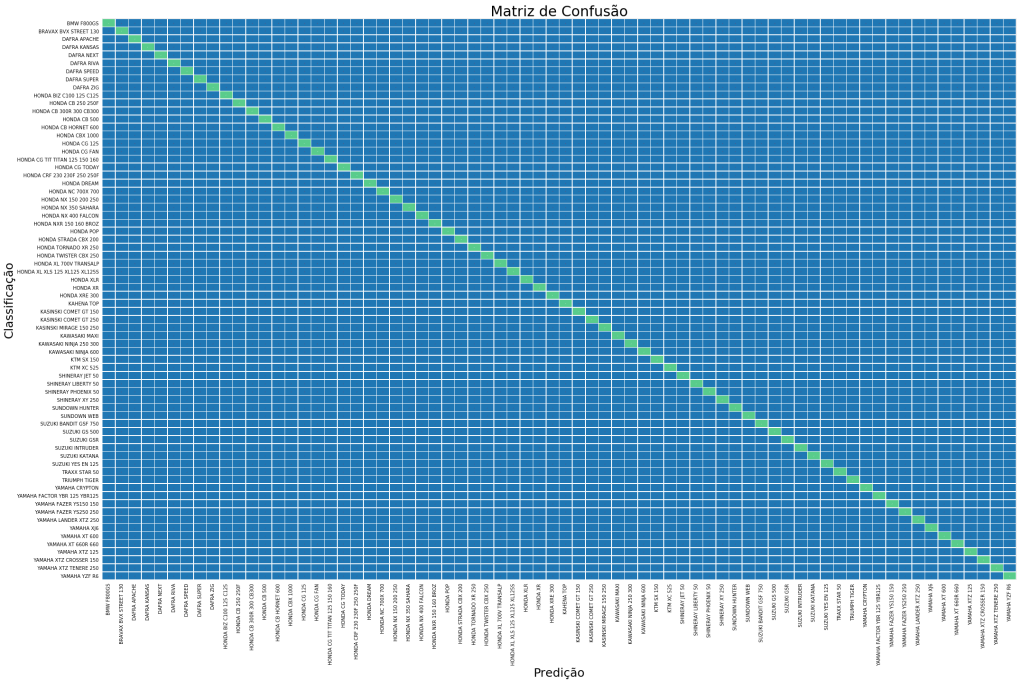
O alerta ao finial do relatório é esperado, tendo em vista que existem aplicações raras e que não aprecerem no grupo de teste.

Matriz de Confusão

In [42]:

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

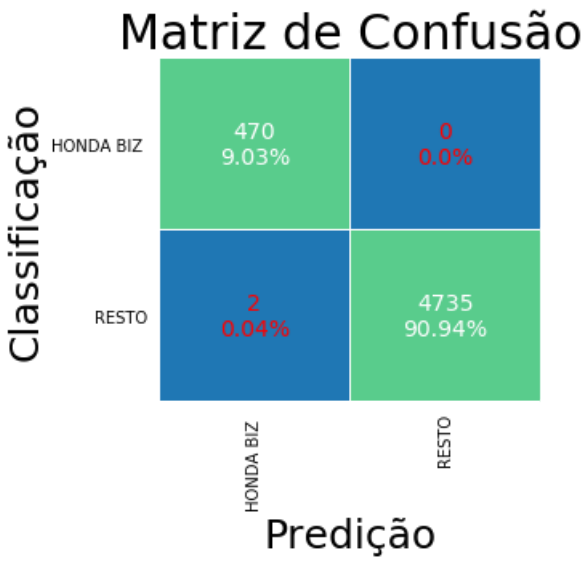
In [43]:
```
pretty_plot_confusion_matrix(dfmclgr, annot=True, cmap="tab10", fmt='.0f', fz=1, lw
=0.5, cbar=False,
                             figsize=[30,20], show_null_values=2, pred_val_axis='x'
,insertTot=False)
```



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 [44]:
```
# gera o plot da matriz de confusão OvR para o modtste do resultadosvc
modteste=dfaplicacoes['APLICACOES'].iloc[276]
plotConfusaoOvR(modteste, aplicacoeslgr, cmlgr)
```



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 [45]:
```
print('index:', linha)
descricao = df.iloc[linha]['DESCRICAO DO PRODUTO']
# verifica a descrição do produto
descricao
```

```
index: 15
```

Out[45]:
```
'80372 KIT DE TRANSMISSÃO, COMPOSTO DE CORRENTE, COROA E PINHÃO PARA MOTOCICLETA R
IVA150 DAFRA, MARCA ALLEN.'
```

In [46]:
```
classificaAplicacaoLGR(criaModelo(descricao))
```

Out[46]:
```
'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 reutilição 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 [47]:
```
# 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 [48]:
```
dfABIMOTO = pd.read_excel(r'./bases/dfABIMOTOv13.xlsx')
```

In [49]:
```
dfABIMOTO.sample()
```

Out[49]:

	PARTES E PEÇAS	VMLE	RETENTOR	APLICACAO
6	KIT TRANSMISSÃO 1045 TITAN 2000/04 14D+428HX11...	7.5	True	HONDA CG TIT TITAN 125 150 160

Classificando a Tabela ABIMOTO

In [50]:
```
dfABIMOTO['APLICACAO']=dfABIMOTO['PARTES E PEÇAS'].apply(classificaAplicacaoSVC)
```

In [51]:
```
dfABIMOTO.sample()
```

Out[51]:

	PARTES E PEÇAS	VMLE	RETENTOR	APLICACAO
1	KIT TRANSMISSÃO 1045 C100 BIZ 15D+428HX108+35D...	3.9	False	HONDA BIZ C100 125 C125

Exportando o DataSet Classificado

Exportando para um arquivo CSV

In [52]:
```
dfABIMOTO.to_csv(r'./bases/dfABIMOTOv13.csv', index = False)
```

Exportando para um arquivo de planilha do Excel

In [53]:
```
dfABIMOTO.to_excel(r'./bases/dfABIMOTOv13.xlsx', index = False, header = True)
```

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: 51.27 segundos.
```