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, TfidfTransform
er
# 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, acresce
ntaMarca, retentorAux, classificaAplicacaoSVC
# Variávis: 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.localt
ime(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(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	Mode
10427	CHINA, REPUBLICA POP	Ý10822/i45¨ 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 carg cg fa honc
12511	CHINA, REPUBLICA POP	item 22;Partes e peças para Motocicletas,Desta...	4.9450	ybr 125 factor	12 fact yamah y
6179	CHINA, REPUBLICA POP	22376 - 71899 - KIT DE TRANSMISSAO 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 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_si
ze=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=T
rue,
     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(mod
elo))
```

```
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, resultadosv
c,average='micro')))
print('Recall:   {:.2f}'.format(metrics.recall_score(y_test, resultadosvc,a
verage='micro')))
print('F1_Score: {:.2f}'.format(metrics.f1_score(y_test, resultadosvc,avera
ge='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))
```

	precision	recall	f1-score	support
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				
	KAHENA 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				

SHINERAY XY 250	1.00	1.00	1.00	4
SUNDOWN HUNTER	1.00	1.00	1.00	6
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\cla
ssification.py:1143: UndefinedMetricWarning: Precision and F-s
core are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
```

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]:

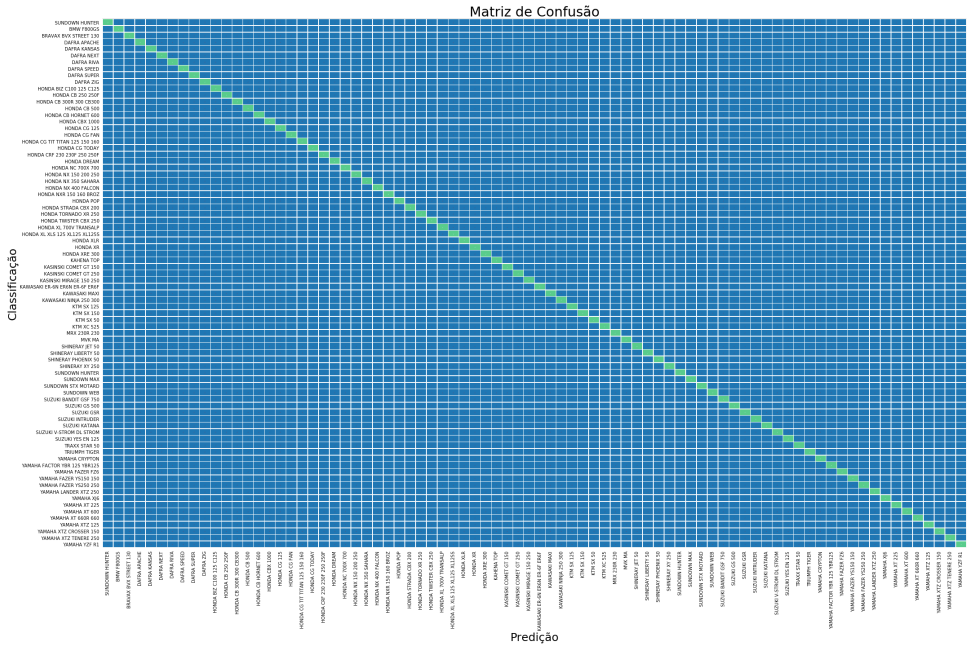
```
# importa os módulos
# gera a matriz de confusão formatada (adaptado por mim)
from prettyPlotConfusionMatrix import pretty_plot_confusion_matrix # pretty
PlotConfusionMatrix.py local
# código disponível em https://github.com/wcipriano/pretty-print-confusion-
matrix
from sklearn.metrics import confusion_matrix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

In [19]:

```
aplicacoessvc = np.unique(resultadosvc) # define os aplicações presentes no
resultado
cmsvc = confusion_matrix(y_test, resultadosvc, aplicacoessvc) # cria a matr
iz de confusão
dfmcsvc=pd.DataFrame(cmsvc,index=aplicacoessvc,columns=aplicacoessvc) # con
verte 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.

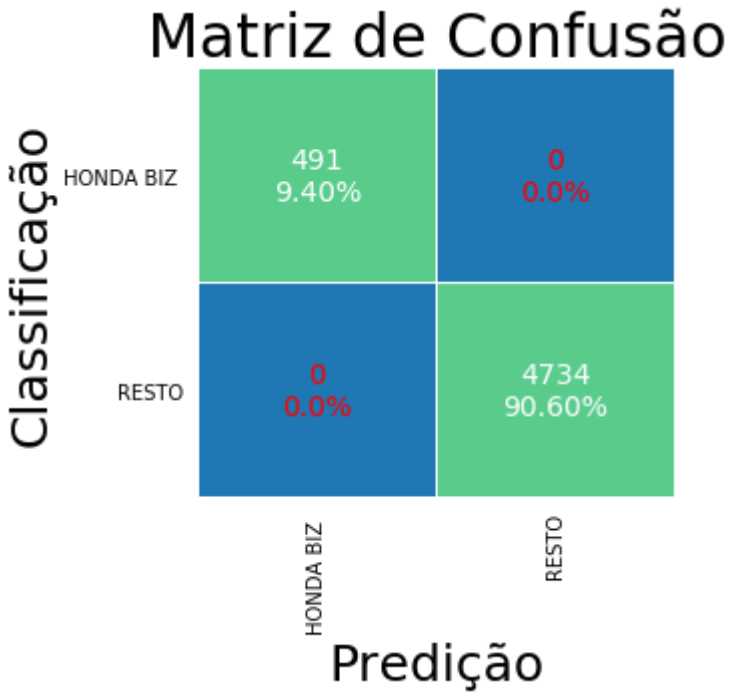
Para melhorar a visão, esboçaremos a matriz de confusão para o item 'HONDA BIZ 100 C100 125 C125' versus o resto (OVR), isso nos permitirá ver a classificação como se fosse uma classificação binária OvR do tipo ou é a classificação demonstrada ou é o resto.

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 re
sultado
    pos = int(np.where(aplicacoes == modelo)[0]) # econtra a posição de mod
elo 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, figs
ize=[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, resultadomn
b,average='micro')))
print('Recall:   {:.2f}'.format(metrics.recall_score(y_test, resultadomnb,a
verage='micro')))
print('F1_Score: {:.2f}'.format(metrics.f1_score(y_test, resultadomnb,avera
ge='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))
```

	precision	recall	f1-score	support
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				
	KAHENA 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\cla
ssification.py:1143: UndefinedMetricWarning: Precision and F-s
core are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cla
ssification.py:1143: UndefinedMetricWarning: Precision and F-s
core are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cla
ssification.py:1143: UndefinedMetricWarning: Precision and F-s
core are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
```

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

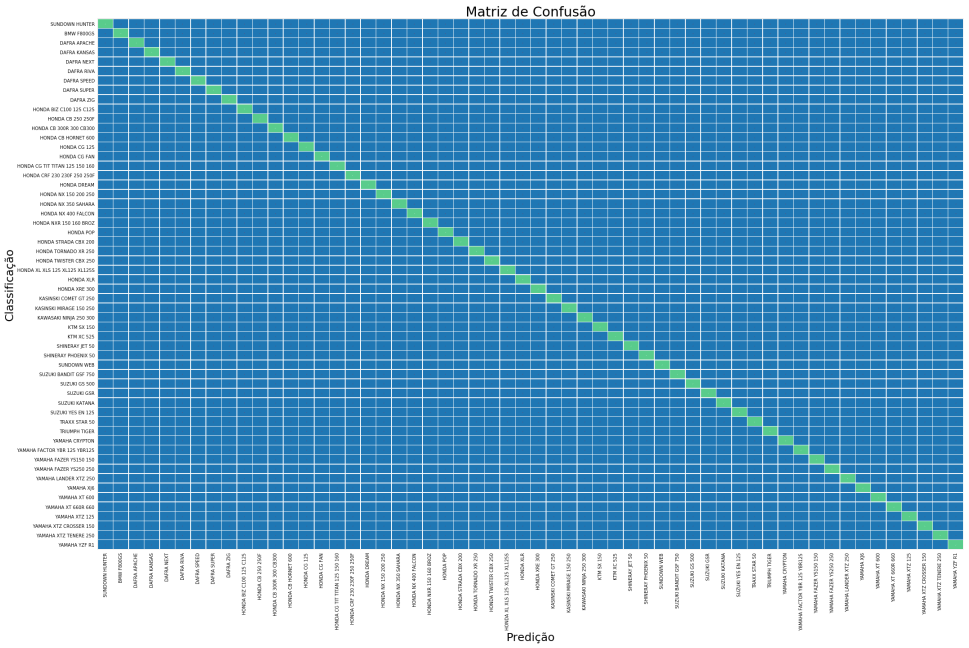
Matriz de Confusão

In [31]:

```
aplicacoesmnb = np.unique(resultadomnb) # define os aplicações presentes no resultado
cmmnb = confusion_matrix(y_test, resultadomnb, aplicacoesmnb) # cria a matriz 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)
```

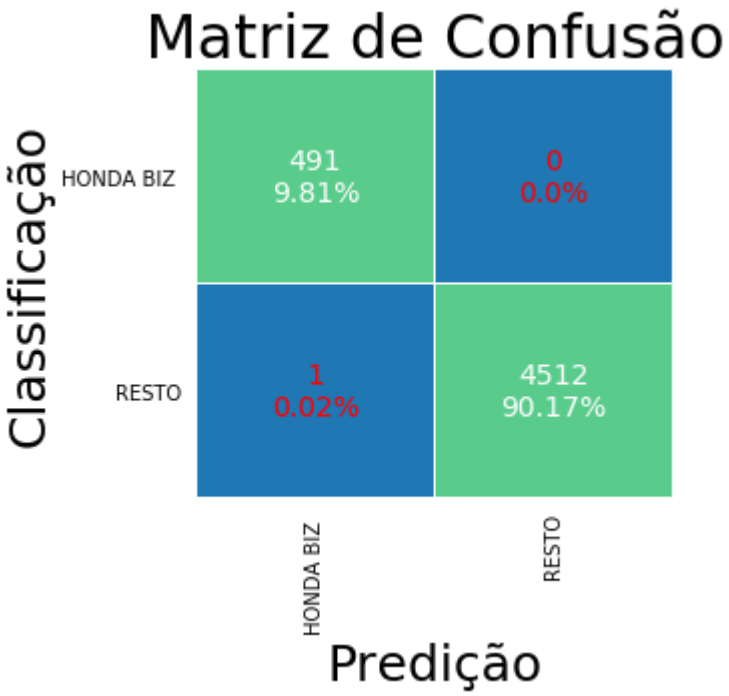


Apesar do grande número de classes, pode-se observar a linha formada pelos acertos na diagonal da imagem, demonstrando o alto índice de acerto do modelo.

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 PINHÃ
O 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_i
ntercept=True,
          intercept_scaling=1, max_iter=100, multi_class='mult
inomial',
          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, resultadolg
r,average='micro')))
print('Recall:   {:.2f}'.format(metrics.recall_score(y_test, resultadolgr,a
verage='micro')))
print('F1_Score: {:.2f}'.format(metrics.f1_score(y_test, resultadolgr,avera
ge='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))
```

	precision	recall	f1-score	support
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
KAHENA 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

SHINERAY XY 250	1.00	1.00	1.00	4
SUNDOWN HUNTER	1.00	1.00	1.00	6
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\cla
ssification.py:1143: UndefinedMetricWarning: Precision and F-s
core are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cla
ssification.py:1143: UndefinedMetricWarning: Precision and F-s
core are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cla
ssification.py:1143: UndefinedMetricWarning: Precision and F-s
core are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
  'precision', 'predicted', average, warn_for)
```

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]:

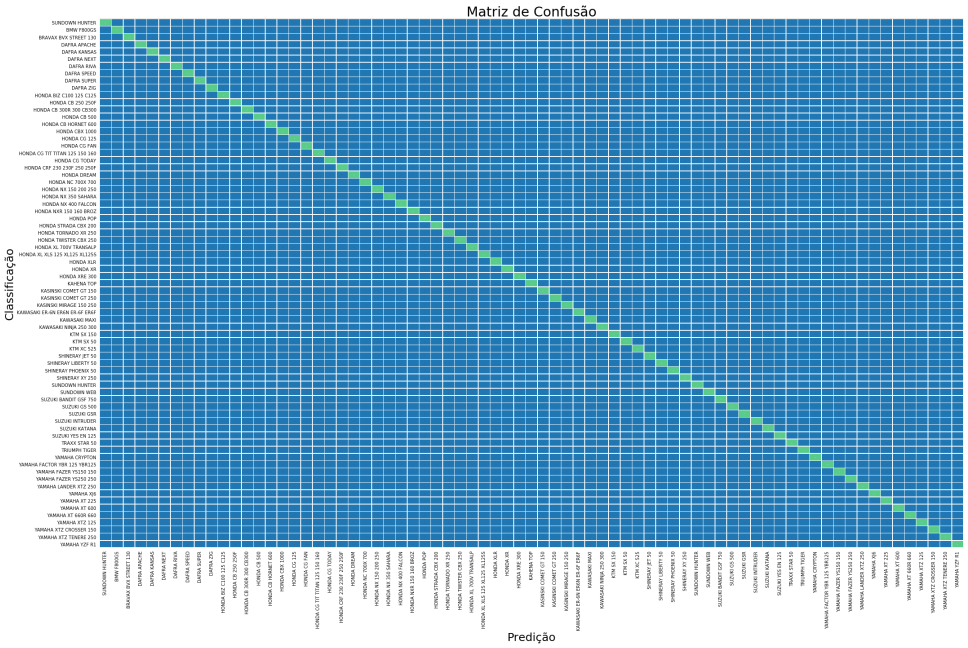
```
# importa os módulos
# gera a matriz de confusão formatada (adaptado por mim)
from prettyPlotConfusionMatrix import pretty_plot_confusion_matrix # pretty
PlotConfusionMatrix.py local
# código disponível em https://github.com/wcipriano/pretty-print-confusion-
matrix
from sklearn.metrics import confusion_matrix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

In [43]:

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

In [44]:

```
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)
```

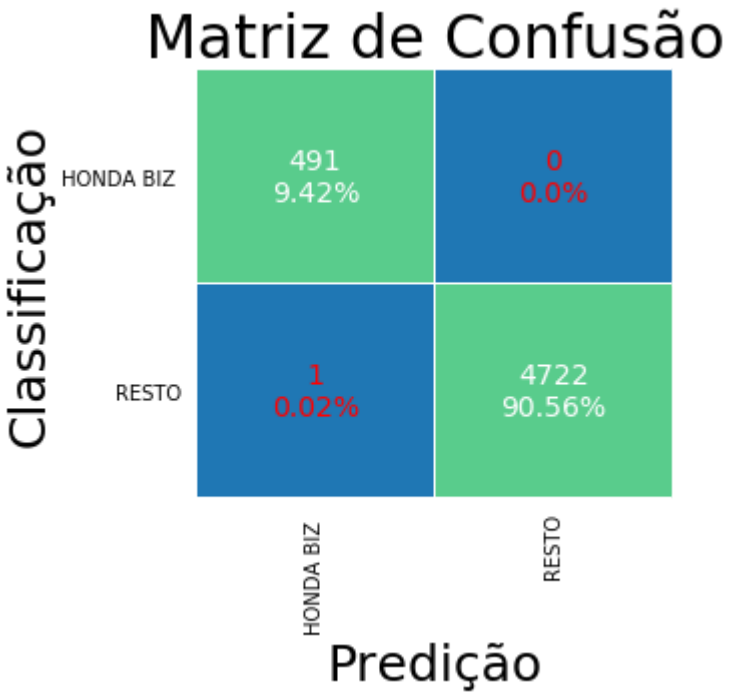


Apesar do grande número de classes, pode-se observar a linha formada pelos acertos na diagonal da imagem, demonstrando o alto índice de acerto do modelo.

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 [45]:

```
# 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 [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 PINHÃ
O 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 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 [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/dfABIMOTOv13.xlsx', index = False, header = True
)
```

Exportando para um arquivo de planilha do Excel

In [54]:

```
dfABIMOTO.to_excel(r'./bases/dfABIMOTOv13.xlsx', index = False, header = Tr
ue)
```

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.
```