Analise de Base Íris

Iremos realizar uma analise de distribuição de classes:

- Utilizaremos o método k-meas

In [7]:

```
# Para suportar python 2 e python 3:
from __future__ import division, print_function, unicode_literals

# Importações Importantes ao projeto:
import numpy as np
from sklearn import datasets
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [4]:

```
# Para traçar belas figuras
plt.rc('axes', labelsize=14)
plt.rc('xtick', labelsize=12)
plt.rc('ytick', labelsize=12)

# para tornar a saída deste notebook estável em todas as execuções
np.random.seed(1234)
```

In [5]:

```
#Desativa Avisos:
import warnings
def fxn():
    warnings.warn("deprecated", DeprecationWarning)
with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    fxn()
with warnings.catch_warnings():
    warnings.filterwarnings("ignore")

warnings.filterwarnings(action="ignore", message="^internal gelsd")
```

In [9]:

```
# Monta o caminho para a estrutura de pastas
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "Forum_Unidade_I"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, CHAPTER_ID, "images")
```

In [10]:

```
# Onde salvar as figuras
def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution,bbox_inches='tight')
```

In [11]:

```
#Carregando a base de dados
iris = datasets.load_iris()
```

In [12]:

```
#Define número de classes desejadas
cluster = KMeans(n_clusters = 3)
```

In [13]:

```
#Ao executar esta linha nó efetivamente agrupamos os dados em 3 grupo
cluster.fit(iris.data)
```

Out[13]:

```
KMeans(n_clusters=3)
```

In [16]:

```
#para verificar-mos em qual grupo cada registro está:
previsoes = cluster.labels_
print(previsoes)
```

```
[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 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2
 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2
 2 1]
```

In [18]:

```
#Nós podemos visualizar os centróides
centroides = cluster.cluster_centers_
print(centroides)
```

```
[[5.006      3.428      1.462      0.246     ]
 [5.9016129  2.7483871  4.39354839 1.43387097]
 [6.85       3.07368421 5.74210526 2.07105263]]
```

In [19]:

```
'''
    Aqui nós podemos verificar através da matrix de confusão a acertividade do modelo
'''
resultados = confusion_matrix(iris.target, previsoes)
print(resultados)
```

```
[[50  0  0]
 [ 0 48  2]
 [ 0 14 36]]
```

In [21]:

```
'''
    Vamos plotar de forma gráfica para melhor visualização como os grupos forma distrib
uidos
'''
plt.scatter(iris.data[previsoes == 0, 0], iris.data[previsoes == 0, 3],
            c = 'green', label = 'Setosa')
plt.scatter(iris.data[previsoes == 1, 0], iris.data[previsoes == 1, 3],
            c = 'red', label = 'Versicolor')
plt.scatter(iris.data[previsoes == 2, 0], iris.data[previsoes == 2, 3],
            c = 'blue', label = 'Virgica')
plt.legend()
```

Out[21]:

```
<matplotlib.legend.Legend at 0x227c92775c0>
```