Apresentação

Esse é o meu primeiro modelo de Machine Learning, aplicado no Dataset de Classificação de flores Iris (https://archive.ics.uci.edu/ml/machine-learning-databases/iris/). Foram utilizados dois algoritmos: KNN e Árvore de Decisão. Aqui busquei focar em algumas bibliotecas de visualização e nos processos da análise de dados. Basicamente, esse é o meu 'Hello World' em Machine Learning.

1. Importando bibliotecas e dados

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import neighbors
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score
%matplotlib inline
```

In [2]:

```
header =  ['Comprimento sepala (cm)', 'Largura Sepala (cm)', "Comprimento petala (cm)",
'largura petala(cm)', 'Class']
data = pd.read_csv('iris.data', names=header)
```

2. Manipulação e Visualização dos dados

In [3]:

```
data.head()
```

Out[3]:

	Comprimento sepala (cm)	Largura Sepala (cm)	Comprimento petala (cm)	largura petala(cm)	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:

```
data['Class'].unique()
```

Out[4]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [5]:

```
def getDummies(x):
    if x == 'Iris-setosa':
        return 0
    elif x == 'Iris-versicolor':
        return 1
    else:    #'Iris-virginica'
        return 2
```

In [6]:

```
data['Class_dummies'] = data['Class'].map(getDummies)
```

In [7]:

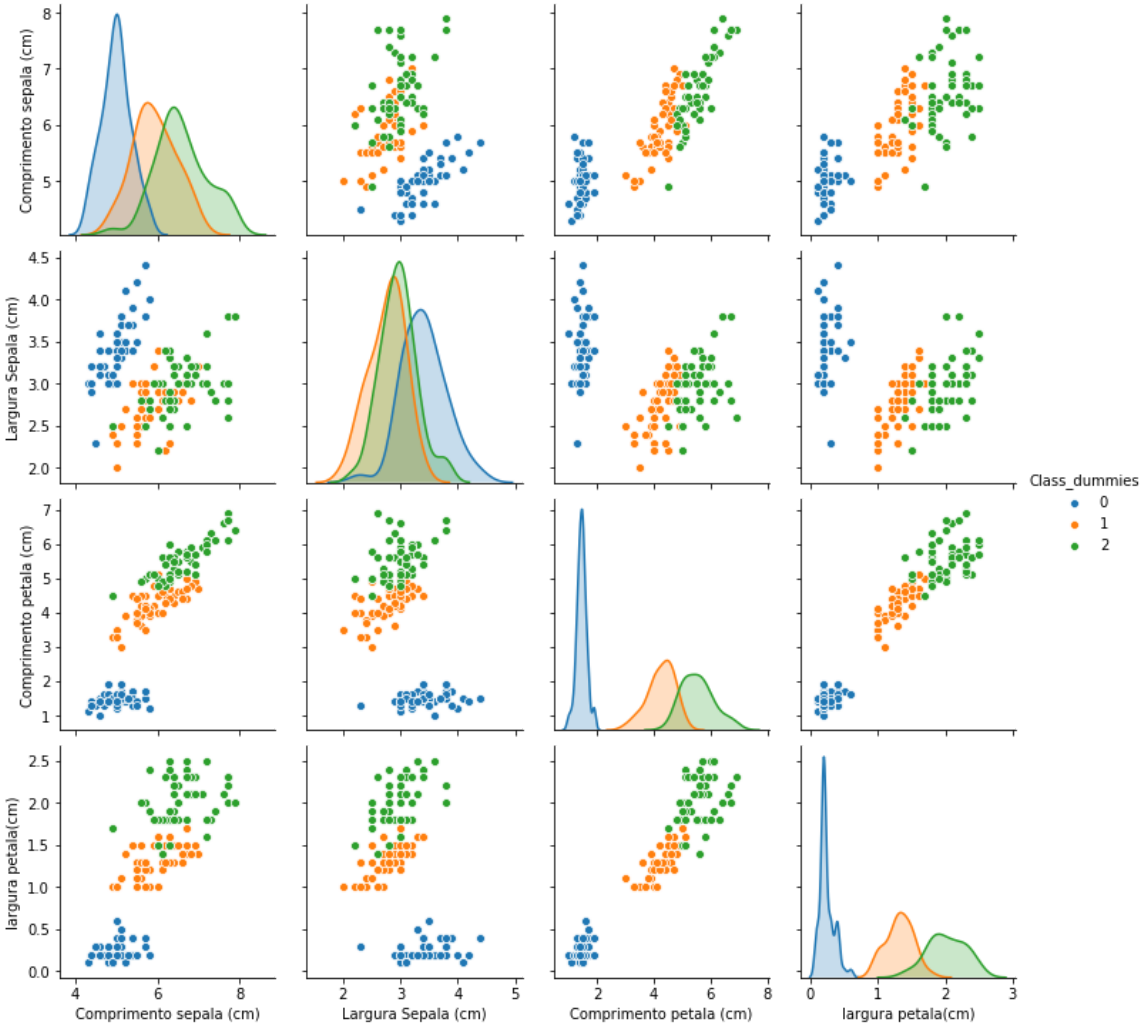
```
data.drop('Class', axis=1, inplace=True)
```

In [8]:

```
sns.pairplot(data, hue="Class_dummies")
#{0:'Iris-setosa', 1:'Iris-versicolor', 2:'Iris-virginica'}
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x25ed2739f48>
```



3. Criação do Modelo

3.1 Separação Treino e Teste

In [9]:

```
X = data.drop('Class_dummies', axis=1)
y = data['Class_dummies']
```

In [10]:

```
X_treino, X_teste, y_treino, y_teste = train_test_split(X, y, test_size=0.3)
```

3.2 KNN

In [11]:

```
modelknn = neighbors.KNeighborsClassifier(3) #nº de vizinhos igual a 3
```

In [12]:

```
modelknn.fit(X_treino,y_treino)
```

Out[12]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

In [13]:

```
modelknn.predict(X_teste)
```

Out[13]:

```
array([2, 2, 2, 1, 1, 0, 1, 0, 0, 2, 2, 2, 0, 2, 0, 2, 2, 0, 2, 0, 0, 1,
       0, 0, 2, 0, 2, 0, 0, 1, 0, 0, 1, 2, 1, 1, 1, 1, 2, 2, 1, 0, 0, 1,
       1], dtype=int64)
```

In [14]:

```
y_testeDFknn=pd.DataFrame(y_teste)
y_previstoknn=pd.DataFrame(modelknn.predict(X_teste), columns=["Classe_prevista"])
```

In [15]:

```
Contagem_testeknn = pd.DataFrame(y_testeDFknn['Class_dummies'].value_counts())
Contagem_previstoknn = pd.DataFrame(y_previstoknn['Classe_prevista'].value_counts())
```

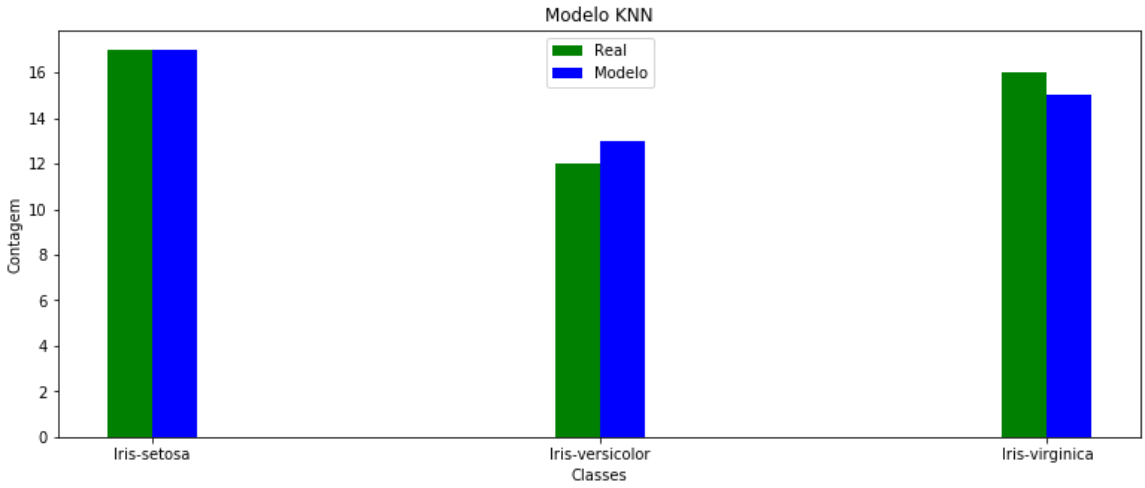
In [16]:

```
fig = plt.figure(figsize=(13,5))
ax = fig.add_subplot(111)
ind = np.arange(3)
ax.bar(Contagem_testeknn.index-0.05,height=Contagem_testeknn['Class_dummies'],width=0.1
,color='g',label='Real')
ax.bar(Contagem_previstoknn.index+0.05,height=Contagem_previstoknn['Classe_prevista'],w
idth=0.1,color='b',label='Modelo')
ax.set_xlabel('Classes')
ax.set_ylabel('Contagem')
ax.set_title('Modelo KNN')
plt.xticks([0,1,2],('Iris-setosa','Iris-versicolor','Iris-virginica'))

plt.legend()
```

Out[16]:

```
<matplotlib.legend.Legend at 0x25ed37dc188>
```



3.3 Decision Tree

In [17]:

```
modelDT = DecisionTreeClassifier()
```

In [18]:

```
modelDT.fit(X_treino,y_treino)
```

Out[18]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [19]:

```
modelDT.predict(X_teste)
```

Out[19]:

```
array([2, 2, 2, 2, 1, 0, 2, 0, 0, 2, 2, 2, 0, 2, 0, 2, 2, 0, 2, 0, 0, 1,
       0, 0, 2, 0, 2, 0, 0, 1, 0, 0, 1, 2, 1, 1, 1, 1, 2, 2, 1, 0, 0, 1,
       1], dtype=int64)
```

In [20]:

```
y_testeDFDT=pd.DataFrame(y_teste)
y_previstoDT=pd.DataFrame(modelDT.predict(X_teste), columns=["Classe_prevista"])
```

In [21]:

```
Contagem_testeDT = pd.DataFrame(y_testeDFDT['Class_dummies'].value_counts())
Contagem_previstoDT = pd.DataFrame(y_previstoDT['Classe_prevista'].value_counts())
```

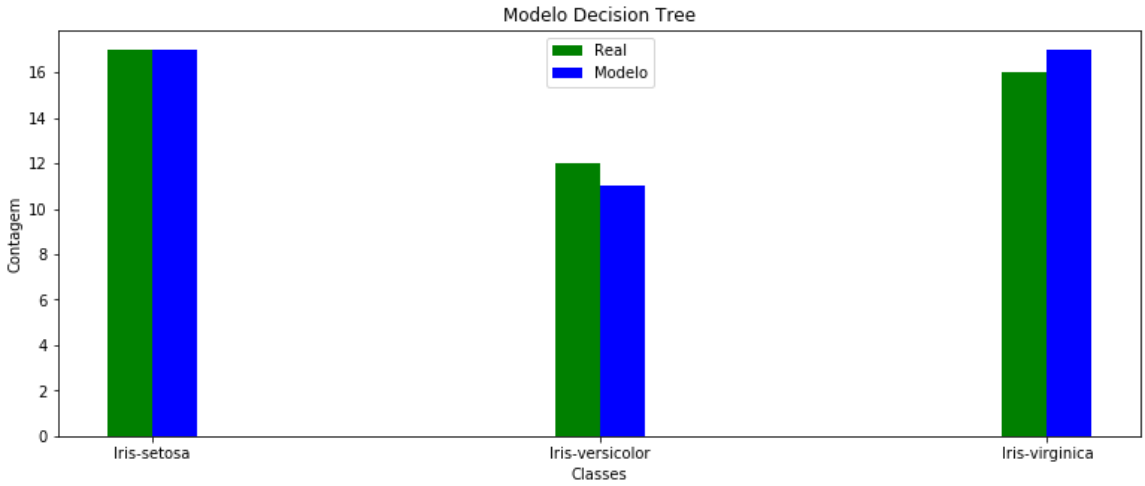
In [22]:

```
fig = plt.figure(figsize=(13,5))
ax = fig.add_subplot(111)
ind = np.arange(3)
ax.bar(Contagem_testeDT.index-0.05,height=Contagem_testeDT['Class_dummies'],width=0.1,c
olor='g',label='Real')
ax.bar(Contagem_previstoDT.index+0.05,height=Contagem_previstoDT['Classe_prevista'],wid
th=0.1,color='b',label='Modelo')
ax.set_xlabel('Classes')
ax.set_ylabel('Contagem')
ax.set_title('Modelo Decision Tree')
plt.xticks([0,1,2],('Iris-setosa','Iris-versicolor','Iris-virginica'))

plt.legend()
```

Out[22]:

```
<matplotlib.legend.Legend at 0x25ed39adc08>
```



4. Validação dos modelos

In [23]:

```
knn = round((np.mean(y_teste==modelknn.predict(X_teste))*100),2)

DT = round((np.mean(y_teste==modelDT.predict(X_teste))*100),2)

print("O modelo KNN acertou {}% dos dados de teste".format(knn))
print()
print("O modelo Decision Tree acertou {}% dos dados de teste".format(DT))
```

```
O modelo KNN acertou 97.78% dos dados de teste

O modelo Decision Tree acertou 97.78% dos dados de teste
```

4.1 Validação Cruzada

In [24]:

```
round(np.mean(cross_val_score(modelknn, X,y, cv=10)),2)
```

Out[24]:

```
0.97
```

In [25]:

```
round(np.mean(cross_val_score(modelDT, X,y, cv=10)),2)
```

Out[25]:

```
0.95
```