Support Vector Machine

Professor: Victor Venites

Data: 14/04/2020

Aula: 07

Exercício feito pelo aluno: Thiago Moura

1. Importando as Bibliotecas de Python

- E Verificando o Ambiente

In [3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

- Print Working Directory

In [4]:

```
pwd
```

Out[4]:

```
'C:\\Users\\thiag\\Documents\\Aulas-Python\\School of AI\\Ano de 2020\\07_
aula'
```

2 - Listando diretório dentro da pasta

In [5]:

```
!dir
```

```
 O volume na unidade C não tem nome.
 O Número de Série do Volume é 3A4A-84B2

 Pasta de C:\Users\thiag\Documents\Aulas-Python\School of AI\Ano de 2020\0
7_aula

22/04/2020  18:55    <DIR>          .
22/04/2020  18:55    <DIR>          ..
15/04/2020  16:11    <DIR>          .ipynb_checkpoints
22/04/2020  18:55           160.052 2020.Aula07_SVM_Prof.VictorVenites_Esc
olaLivreIA_Python.ipynb
14/04/2020  18:57         1.503.515 2020.Aula07_SVM_VictorVenites_EscolaLi
vreIA_Slides.pdf
14/04/2020  18:57             8.945 AND.xlsx
14/04/2020  18:57            42.388 breastCancer.xlsx
14/04/2020  20:02            13.894 Gráfico_AND.jpg
14/04/2020  20:00            13.757 Gráfico_OR.jpg
14/04/2020  20:03            13.792 Gráfico_XOR.jpg
14/04/2020  18:57            13.894 Gr†fico_AND.jpg
14/04/2020  18:57            13.757 Gr†fico_OR.jpg
14/04/2020  18:57            13.792 Gr†fico_XOR.jpg
14/04/2020  18:57             8.955 OR.xlsx
15/04/2020  16:11                72 Untitled.ipynb
14/04/2020  18:57             9.103 XOR_Kernel.xlsx
              13 arquivo(s)      1.815.916 bytes
               3 pasta(s)   71.400.730.624 bytes disponíveis
```

- Minha Versão do Python
- Python 3.7.3

In [6]:

```
!python  --version
```

```
Python 3.7.3
```

2. Base de Dados OR

- Verdadeiro OU Falso

In [7]:

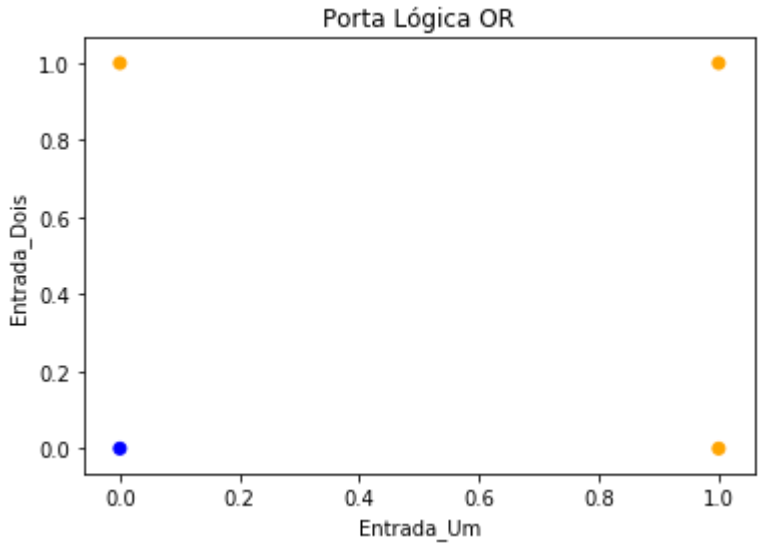
```
df_OR = pd.read_excel("OR.xlsx", index_col = "Lição")
df_OR
```

Out[7]:

Lição	Entrada_Um	Entrada_Dois	Saida_OR
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

In [8]:

```
cores = ["blue", "orange", "orange", "orange"]
plt.scatter(df_OR["Entrada_Um"], df_OR["Entrada_Dois"], c = cores)
plt.title("Porta Lógica OR")
plt.xlabel("Entrada_Um")
plt.ylabel("Entrada_Dois")
plt.savefig("Gráfico_OR.jpg")
plt.show()
```



3. Base de Dados AND

- Verdadeiro E Falso

In [9]:

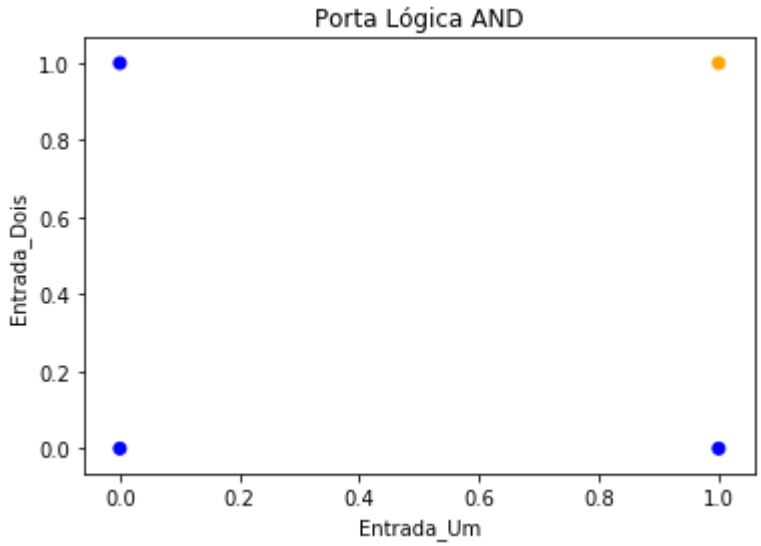
```
df_AND = pd.read_excel("AND.xlsx", index_col = "Lição")
df_AND
```

Out[9]:

Lição	Entrada_Um	Entrada_Dois	Saida_AND
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

In [7]:

```
cores = ["blue", "blue", "blue", "orange"]
plt.scatter(df_AND["Entrada_Um"], df_AND["Entrada_Dois"], c = cores)
plt.title("Porta Lógica AND")
plt.xlabel("Entrada_Um")
plt.ylabel("Entrada_Dois")
plt.savefig("Gráfico_AND.jpg")
plt.show()
```



4. Vamos Criar Nosso "BigData" de XOR

In [10]:

```
Entrada_Um = [0, 0, 1, 1]
Entrada_Dois = [0, 1, 0, 1]
Saida_XOR = [0, 1, 1, 0]
Colunas = ["Entrada_Um", "Entrada_Dois", "Saida_XOR"]
Indice = [1, 2, 3, 4]
```

In [11]:

```
df_XOR = pd.DataFrame([], columns = Colunas, index = Indice)
df_XOR
```

Out[11]:

	Entrada_Um	Entrada_Dois	Saida_XOR
1	NaN	NaN	NaN
2	NaN	NaN	NaN
3	NaN	NaN	NaN
4	NaN	NaN	NaN

In [12]:

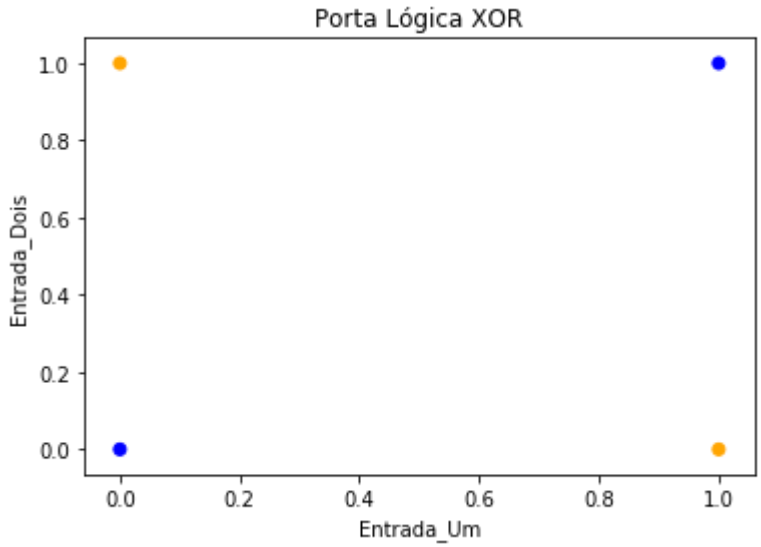
```
df_XOR["Entrada_Um"] = Entrada_Um
df_XOR["Entrada_Dois"] = Entrada_Dois
df_XOR["Saida_XOR"] = Saida_XOR
df_XOR
```

Out[12]:

	Entrada_Um	Entrada_Dois	Saida_XOR
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

In [13]:

```
cores = ["blue", "orange", "orange", "blue"]
plt.scatter(df_XOR["Entrada_Um"], df_XOR["Entrada_Dois"], c = cores)
plt.title("Porta Lógica XOR")
plt.xlabel("Entrada_Um")
plt.ylabel("Entrada_Dois")
plt.savefig("Gráfico_XOR.jpg")
plt.show()
```



5. Modelo Support Vector Machine

- Machina de Vetor de Suporte

Uma linha de comando

In [15]:

```
df_OR
```

Out[15]:

Lição	Entrada_Um	Entrada_Dois	Saida_OR
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

5.2. Versão do SkLearn

In [16]:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
from sklearn.metrics import confusion_matrix
```

5.2.1. Classificando OR

In [17]:

```
X_OR = df_OR.iloc[:, 0:-1]
y_OR = df_OR.iloc[:, -1]
y_OR
```

Out[17]:

```
Lição
1    0
2    1
3    1
4    1
Name: Saida_OR, dtype: int64
```

In [18]:

```
svm_clf_OR = Pipeline([
    ("scaler", StandardScaler()),
    ("linear_svc", LinearSVC(C = 0.5, loss = "hinge", penalty='l2')),
])
svm_clf_OR.fit(X_OR, y_OR)
```

Out[18]:

```
Pipeline(memory=None,
         steps=[('scaler',
                 StandardScaler(copy=True, with_mean=True, with_std=Tru
e)),
                ('linear_svc',
                 LinearSVC(C=0.5, class_weight=None, dual=True,
                           fit_intercept=True, intercept_scaling=1,
                           loss='hinge', max_iter=1000, multi_class='ovr',
                           penalty='l2', random_state=None, tol=0.0001,
                           verbose=0))],
         verbose=False)
```

5.2.1.1. Resultados OR

In [19]:

```
svm_clf_OR.predict([[0, 0]])
```

Out[19]:

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

In [20]:

```
svm_clf_OR.predict([[0, 1]])
```

Out[20]:

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

In [21]:

```
svm_clf_OR.predict([[1, 0]])
```

Out[21]:

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

In [22]:

```
svm_clf_OR.predict([[1, 1]])
```

Out[22]:

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

5.2.1.2. Matriz de Confusão

- A função consider os rótulos na ordem de chegada
- 0 como sendo a primeira classificação
- 1 como sendo a segunda classificação

In [23]:

```
Predicao_OR = svm_clf_OR.predict(X_OR)
confusion_matrix(y_OR, Predicao_OR)
```

Out[23]:

```
array([[1, 0],
       [0, 3]], dtype=int64)
```

In [24]:

```
df_OR_Predicao = df_OR.copy()
df_OR_Predicao["Predicao"] = Predicao_OR
df_OR_Predicao
```

Out[24]:

Lição	Entrada_Um	Entrada_Dois	Saida_OR	Predicao
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	1	1

5.2.1.1. Plotando o Vetor de Suporte

- mlxtend
- Pacote para imprimir Gráficos de Classificação bonitos, tipo nos livros
- Herda do MatPlotLib
- pip ==> Pacote de instalação do Python

In [25]:

```
!pip install mlxtend
```

```
Requirement already satisfied: mlxtend in c:\users\thiag\anaconda3\lib\sit
e-packages (0.17.2)
Requirement already satisfied: scipy>=1.2.1 in c:\users\thiag\anaconda3\li
b\site-packages (from mlxtend) (1.2.1)
Requirement already satisfied: joblib>=0.13.2 in c:\users\thiag\anaconda3
\lib\site-packages (from mlxtend) (0.13.2)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\thiag\anacond
a3\lib\site-packages (from mlxtend) (3.1.0)
Requirement already satisfied: setuptools in c:\users\thiag\anaconda3\lib
\site-packages (from mlxtend) (45.2.0)
Requirement already satisfied: numpy>=1.16.2 in c:\users\thiag\anaconda3\l
ib\site-packages (from mlxtend) (1.16.4)
Requirement already satisfied: scikit-learn>=0.20.3 in c:\users\thiag\anac
onda3\lib\site-packages (from mlxtend) (0.21.2)
Requirement already satisfied: pandas>=0.24.2 in c:\users\thiag\anaconda3
\lib\site-packages (from mlxtend) (0.24.2)
Requirement already satisfied: cycler>=0.10 in c:\users\thiag\anaconda3\li
b\site-packages (from matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\thiag\anacond
a3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\thiag\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxten
d) (2.4.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\thiag\anac
onda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.0)
Requirement already satisfied: pytz>=2011k in c:\users\thiag\anaconda3\lib
\site-packages (from pandas>=0.24.2->mlxtend) (2019.1)
Requirement already satisfied: six in c:\users\thiag\anaconda3\lib\site-pa
ckages (from cycler>=0.10->matplotlib>=3.0.0->mlxtend) (1.12.0)
```

In [26]:

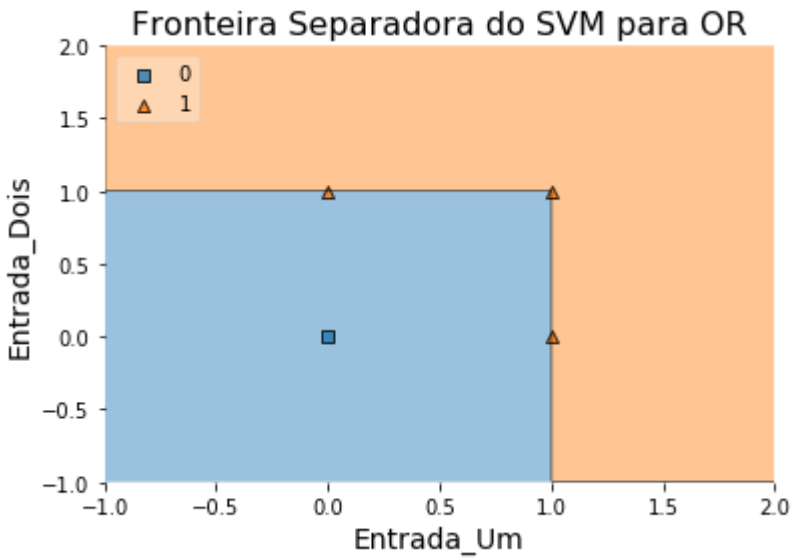
```
from mlxtend.plotting import plot_decision_regions
```

In [27]:

```
Parametros_Plot_mlxtend = {"X" : X_OR.values ,
                           "y" : y_OR.values ,
                           "clf" : svm_clf_OR ,
                           "legend" : 2
                          }
plot_decision_regions(**Parametros_Plot_mlxtend)
plt.xlabel(X_OR.columns[0], size=14)
plt.ylabel(X_OR.columns[1], size=14)
plt.title('Fronteira Separadora do SVM para OR', size=16)
```

Out[27]:

```
Text(0.5, 1.0, 'Fronteira Separadora do SVM para OR')
```



5.2.2. Classificando AND

In [28]:

```
X_AND = df_AND.iloc[:, 0:-1]
y_AND = df_AND.iloc[:, -1]
svm_clf_AND = Pipeline([
    ("scaler", StandardScaler()),
    ("linear_svc", LinearSVC(C = 1, loss = "hinge", penalty='l2')),
])
svm_clf_AND.fit(X_AND, y_AND)
Predicao_AND = svm_clf_AND.predict(X_AND)
confusion_matrix(y_AND, Predicao_AND)
```

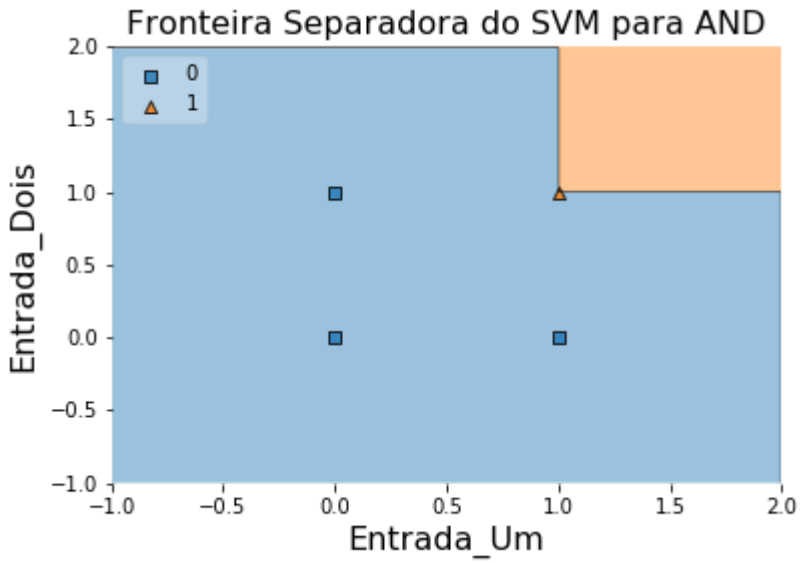
Out[28]:

```
array([[3, 0],
       [0, 1]], dtype=int64)
```

5.2.2.1. Plotando o Vetor de Suporte

In [31]:

```
Parametros_Plot_mlxtend = {"X" : X_AND.values ,
                           "y" : y_AND.values ,
                           "clf" : svm_clf_AND ,
                           "legend" : 2
                          }
plot_decision_regions(**Parametros_Plot_mlxtend)
plt.xlabel(X_AND.columns[0], size=16)
plt.ylabel(X_AND.columns[1], size=16)
plt.title('Fronteira Separadora do SVM para AND', size=16)
plt.show()
```



5.2.3. Classificando XOR

In [32]:

```
X_XOR = df_XOR.iloc[:, 0:-1]
y_XOR = df_XOR.iloc[:, -1]
svm_clf_XOR = Pipeline([
    ("scaler", StandardScaler()),
    ("linear_svc", LinearSVC(C = 1, loss = "squared_hinge", penalty='l2')),
])
svm_clf_XOR.fit(X_XOR, y_XOR)
Predito_XOR = svm_clf_XOR.predict(X_XOR)
Predito_XOR
```

Out[32]:

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

- Por que não deu certo acima?

R: Por que o XOR é não linear

In [33]:

```
from sklearn.preprocessing import PolynomialFeatures
```

In [34]:

```
X_XOR = df_XOR.iloc[:, 0:-1]
y_XOR = df_XOR.iloc[:, -1]
svm_clf_XOR = Pipeline([
    ("poly_features", PolynomialFeatures(degree = 2)),
    ("linear_svc", LinearSVC(C = 1, loss = "squared_hinge", penalty='l2', )),
])
svm_clf_XOR.fit(X_XOR, y_XOR)
Predito_XOR = svm_clf_XOR.predict(X_XOR)
confusion_matrix(y_XOR, Predito_XOR)
```

Out[34]:

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

In [35]:

```
df_XOR_Resultado = df_XOR.copy()
df_XOR_Resultado["Predito_XOR"] = Predito_XOR
df_XOR_Resultado
```

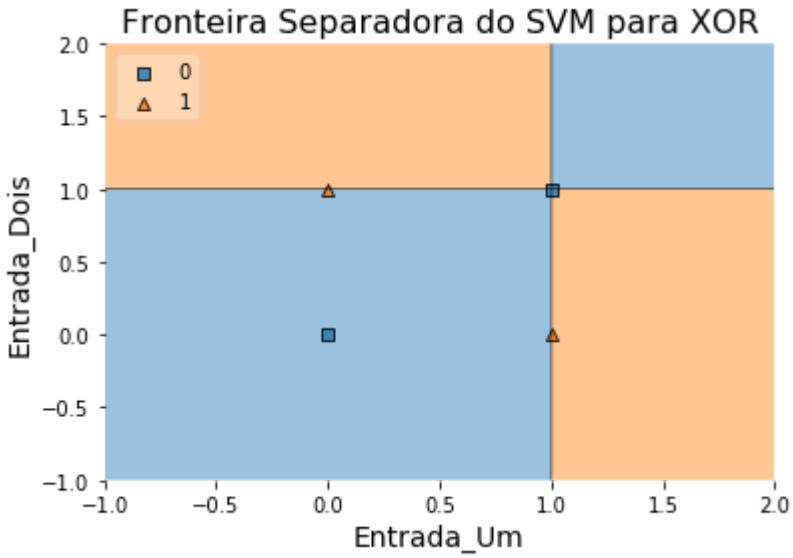
Out[35]:

	Entrada_Um	Entrada_Dois	Saida_XOR	Predito_XOR
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	0	0

5.2.3.1. Plotando o Vetor de Suporte

In [37]:

```
Parametros_Plot_mlxtend = {"X" : X_XOR.values ,
                           "y" : y_XOR.values ,
                           "clf" : svm_clf_XOR ,
                           "legend" : 2
                          }
plot_decision_regions(**Parametros_Plot_mlxtend)
plt.xlabel(X_XOR.columns[0], size=14)
plt.ylabel(X_XOR.columns[1], size=14)
plt.title('Fronteira Separadora do SVM para XOR', size=16)
plt.show()
```



5.2.4. Classificando Tipos de Cancer

In [40]:

```
df_Cancer = pd.read_excel("breastCancer.xlsx", index_col = "id")
df_Cancer.head()
```

Out[40]:

id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_si
1000025	5	1	1	1	
1002945	5	4	4	5	
1015425	3	1	1	1	
1016277	6	8	8	1	
1017023	4	1	1	3	

- Cancer do Tipo 2 ==> Verdadeiro Positivo
- Cancer do Tipo 4 ==> Verdadeiro Negativo

In [56]:

```
X_Cancer = df_Cancer.iloc[:, 0:-1]
y_Cancer = df_Cancer.iloc[:, -1]
svm_clf_Cancer = Pipeline([
    ("scaler", StandardScaler()),
    ("linear_svc", LinearSVC(C = 1, loss = "hinge", penalty='l2', max_iter = 1000)),
])
svm_clf_Cancer.fit(X_Cancer, y_Cancer)
Predicao_Cancer = svm_clf_Cancer.predict(X_Cancer)
confusion_matrix(y_Cancer, Predicao_Cancer)
```

Out[56]:

```
array([[432,  12],
       [  7, 232]], dtype=int64)
```

In [42]:

```
len(y_Cancer)
```

Out[42]:

```
683
```

In [48]:

```
y_Cancer.value_counts()
```

Out[48]:

```
2    444
4    239
Name: class, dtype: int64
```

In [ ]:

```
df_Cancer_Resultado = df_Cancer.copy()
df_Cancer_Resultado["Predicao_Cancer"] = Predicao_Cancer
df_Cancer_Resultado.head()
```

W = X^t *y* + *constante* qsi (qsi é o erro)

- Onde a constante por padrão é igual a 1
- Constante Alta, ou seja, maior que 1, aceita o erro para fazer a correção, considera mais o erro
- Constante Baixa, menor que 1, ou pode ser até 0. Não aceita erros, não quer erro.
- Seguido a constante de Pearson

In [39]:

```
df_Cancer_Resultado.iloc[10:15, -2:]
```

Out[39]:

id	class	Predicao_Cancer
1035283	2	2
1036172	2	2
1041801	4	2
1043999	2	2
1044572	4	4

In [50]:

```
from sklearn.metrics import accuracy_score
```

In [57]:

```
Acuracia_Cancer = accuracy_score(y_Cancer, Predicao_Cancer)
Acuracia_Cancer = round(Acuracia_Cancer * 100, 2)
Acuracia_Cancer
```

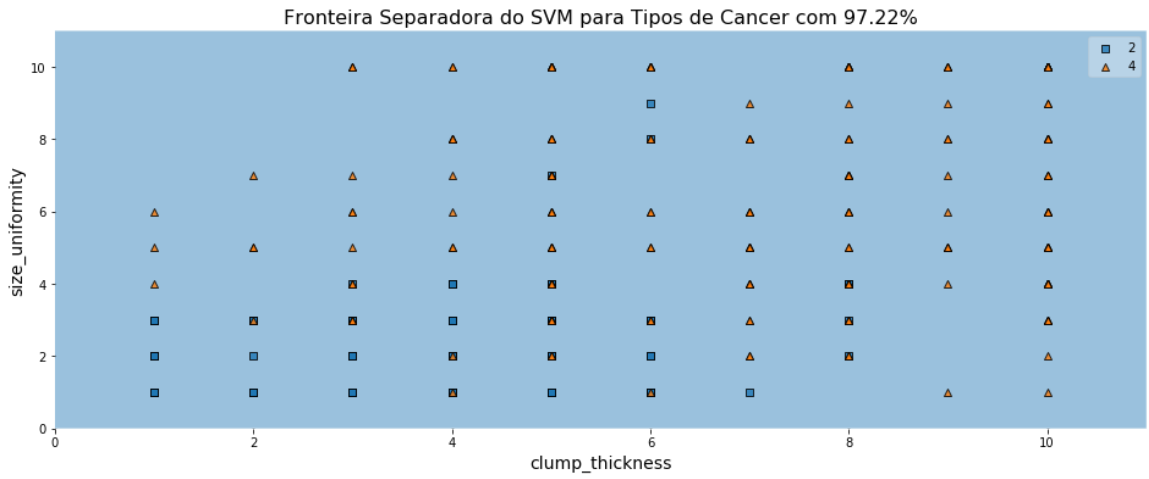
Out[57]:

```
97.22
```

In [58]:

```
valor = 10
plt.figure(figsize = (16, 6))
Parametros_Plot_mlxtend = {"X" : X_Cancer.values,
                           "y" : y_Cancer.values,
                           "clf" : svm_clf_Cancer,
                           "filler_feature_values" : {2: valor, 3: valor, 4: valor, 5:
valor, 6: valor, 7: valor, 8: valor},
                           "filler_feature_ranges" : {2: valor, 3: valor, 4: valor, 5:
valor, 6: valor, 7: valor, 8: valor}
                          }
plot_decision_regions(**Parametros_Plot_mlxtend)
plt.xlabel(X_Cancer.columns[0], size = 14)
plt.ylabel(X_Cancer.columns[1], size = 14)
plt.title(f'Fronteira Separadora do SVM para Tipos de Cancer com {Acuracia_Cancer}%', s
ize = 16)
plt.show()
```

```
C:\Users\thiag\Anaconda3\lib\site-packages\mlxtend\plotting\decision_regio
ns.py:247: UserWarning: No contour levels were found within the data rang
e.
  antialiased=True)
```



In [ ]:

```
### 5.2.5 Rodar a base de Cancer com apenas duas colunas
```

In [43]:

```
df_Cancer.corr().iloc[:, -1:]
```

Out[43]:

	class
clump_thickness	0.714790
size_uniformity	0.820801
shape_uniformity	0.821891
marginal_adhesion	0.706294
epithelial_size	0.690958
bare_nucleoli	0.822696
bland_chromatin	0.758228
normal_nucleoli	0.718677
mitoses	0.423448
class	1.000000

In [45]:

```
df_Cancer.corr().iloc[:,-1:].sort_values(by = "class")
```

Out[45]:

	class
mitoses	0.423448
epithelial_size	0.690958
marginal_adhesion	0.706294
clump_thickness	0.714790
normal_nucleoli	0.718677
bland_chromatin	0.758228
size_uniformity	0.820801
shape_uniformity	0.821891
bare_nucleoli	0.822696
class	1.000000

In [ ]:

```
df_Cancer_Otima = df.Cancer.loc[:,["bare_nucleoli","size_uniformity"]]
```

5.2.5 - Rodar Base de Cancer com apenas duas colunas

In [63]:

```
df_Cancer.corr().iloc[:,-1:].sort_values(by = 'class')
```

Out[63]:

	class
mitoses	0.423448
epithelial_size	0.690958
marginal_adhesion	0.706294
clump_thickness	0.714790
normal_nucleoli	0.718677
bland_chromatin	0.758228
size_uniformity	0.820801
shape_uniformity	0.821891
bare_nucleoli	0.822696
class	1.000000

In [67]:

```
df_Cancer_Otima = df_Cancer.loc[:, ["shape_uniformity", "bare_nucleoli","class"]]
df_Cancer_Otima.head()
```

Out[67]:

id	shape_uniformity	bare_nucleoli	class
1000025	1	1	2
1002945	4	10	2
1015425	1	2	2
1016277	8	4	2
1017023	1	1	2

In [68]:

```
X_Cancer_Otima = df_Cancer_Otima.iloc[:, 0:-1]
y_Cancer_Otima = df_Cancer_Otima.iloc[:, -1]
svm_clf_Cancer_Otima = Pipeline([
    ("scaler", StandardScaler()),
    ("linear_svc", LinearSVC(C = 1, loss = "hinge", penalty='l2', max_iter = 1000)),
])
svm_clf_Cancer_Otima.fit(X_Cancer_Otima, y_Cancer_Otima)
Predicao_Cancer_Otima = svm_clf_Cancer_Otima.predict(X_Cancer_Otima)
confusion_matrix(y_Cancer_Otima, Predicao_Cancer_Otima)
```

Out[68]:

```
array([[430,  14],
       [ 16, 223]], dtype=int64)
```

In [71]:

```
df_Cancer_Otima_Resultado = df_Cancer_Otima.copy()
df_Cancer_Otima_Resultado["Predicao_Cancer"] = Predicao_Cancer_Otima
df_Cancer_Otima_Resultado.head()
```

Out[71]:

id	shape_uniformity	bare_nucleoli	class	Predicao_Cancer
1000025	1	1	2	2
1002945	4	10	2	4
1015425	1	2	2	2
1016277	8	4	2	4
1017023	1	1	2	2

In [72]:

```
Acuracia_Cancer_Otima = accuracy_score(y_Cancer_Otima, Predicao_Cancer_Otima)
Acuracia_Cancer_Otima = round(Acuracia_Cancer_Otima * 100, 2)
Acuracia_Cancer_Otima
```

Out[72]:

```
95.61
```

In [74]:

```
valor = 10
plt.figure(figsize = (16, 6))
Parametros_Plot_mlxtend = {"X" : X_Cancer_Otima.values,
                           "y" : y_Cancer_Otima.values,
                           "clf" : svm_clf_Cancer_Otima,
                           "filler_feature_values" : {2: valor, 3: valor, 4: valor, 5:
valor, 6: valor, 7: valor, 8: valor},
                           "filler_feature_ranges" : {2: valor, 3: valor, 4: valor, 5:
valor, 6: valor, 7: valor, 8: valor}
                          }
plot_decision_regions(**Parametros_Plot_mlxtend)
plt.xlabel(X_Cancer_Otima.columns[0], size = 14)
plt.ylabel(X_Cancer_Otima.columns[1], size = 14)
plt.title(f'Fronteira Separadora do SVM para Tipos de Cancer com {Acuracia_Cancer_Otim
a}%', size = 16)
plt.savefig("Graficos_Cancer_Otima.jpg")
plt.show()
```