IA006 - Aprendizado de Máquina

Exercícios de Fixação de Conceito

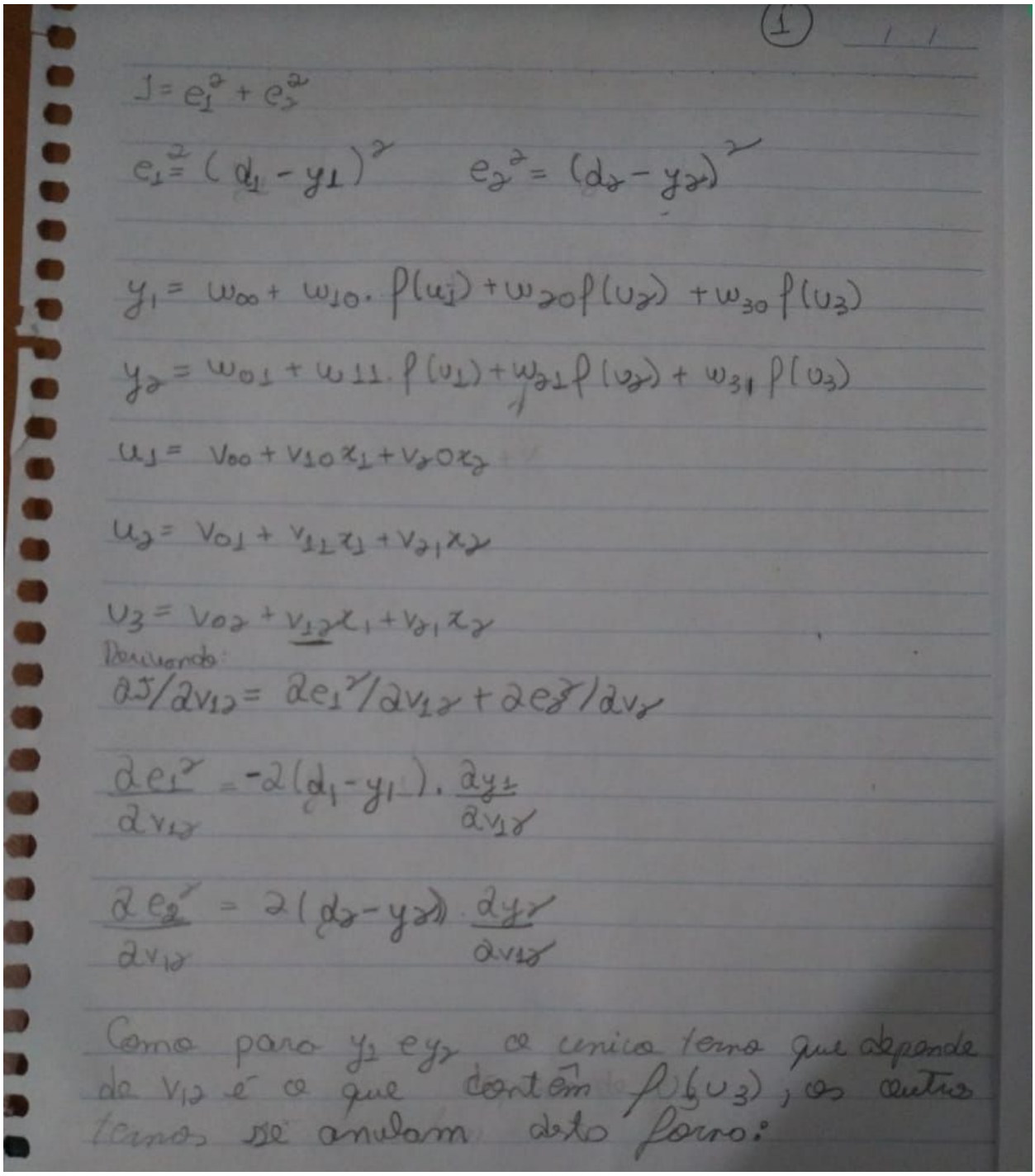
EFC3

Resolução

Nome: *Roger Danilo Figlie* **RA:** *189957*
Nome: *Thiago Arruda Navarro do Amaral* **RA:** *159121*

Profs.: *Levy Boccato e Romis Ribeiro de Faissol Attux*

Parte 1 - Revisitando o algoritmo de retropropagação do erro



(1)

$J = e_1^2 + e_2^2$

$e_1^2 = (d_1 - y_1)^2 \qquad e_2^2 = (d_2 - y_2)^2$

$y_1 = w_{00} + w_{10} \cdot f(u_1) + w_{20} f(u_2) + w_{30} f(u_3)$

$y_2 = w_{01} + w_{11} \cdot f(u_1) + w_{21} f(u_2) + w_{31} f(u_3)$

$u_1 = v_{00} + v_{10} x_1 + v_{20} x_2$

$u_2 = v_{01} + v_{11} x_1 + v_{21} x_2$

$u_3 = v_{02} + v_{12} x_1 + v_{22} x_2$

Derivando:

$\partial J / \partial v_{12} = \partial e_1^2 / \partial v_{12} + \partial e_2^2 / \partial v_{12}$

$\frac{\partial e_1^2}{\partial v_{12}} = -2(d_1 - y_1) \cdot \frac{\partial y_1}{\partial v_{12}}$

$\frac{\partial e_2^2}{\partial v_{12}} = 2(d_2 - y_2) \cdot \frac{\partial y_2}{\partial v_{12}}$

Como para y_1 e y_2 a única terra que depende de v_{12} é a que contém $f(u_3)$, os outros termos se anulam daí para frente:

$$\frac{\partial y_1}{\partial v_{12}} = w_{30} \cdot \frac{\partial f}{\partial v_3} \cdot \frac{\partial v_3}{\partial v_{12}} = w_{30} \cdot \frac{\partial f}{\partial v_3} \cdot x_1$$

$$\frac{\partial y_2}{\partial v_{12}} = w_{31} \cdot \frac{\partial f}{\partial v_3} \cdot \frac{\partial v_3}{\partial v_{12}} = w_{31} \cdot \frac{\partial f}{\partial v_3} \cdot x_1$$

Como v_i nada mais é do que a entrada na nossa função f, temos $\partial f/\partial v_1 = \partial f/\partial v_2 = \partial f/\partial v_3 = \dot{f}$. Daí fomos agregando tudo:

$$\partial J/\partial v_{12} = -2(d_1 - y_1)\,\partial y_1/\partial v_{12} - 2(d_2 - y_2)\,\partial y_2/\partial v_{12}$$

$$\partial J/\partial v_{12} = -2(d_1 - y_1)\,w_{30}\,\dot{f}\,x_1 - 2(d_2 - y_2)\,w_{31}\,\dot{f}\,x_1$$

$$\partial J/\partial v_{12} = -2\dot{f}\,x_1\,[(d_1 - y_1)w_{30} + (d_2 - y_2)w_{31}] \;//$$

Lembrando que y_1 é uma função que depende de todos os pesos, função de ativação e das entradas. E d_i é a variável do sinal do amostra.

Parte 2 - Classificação binária com redes MLP e SVMs

Iniciamos esta parte da lista comentando um pouco sobre a estrutura dos dados. A primeira coisa que atentamos é que aparentemente os atributos já estão standarlizados, pois suas médias são aproximadamente 0,015 e suas variâncias estão próximas de 0,943 (os valores exatos seram explicitados quando tratarmos de SVM's mais abaixo). Ademais verificamos que há pouca diferença na distribuição entre as classes:

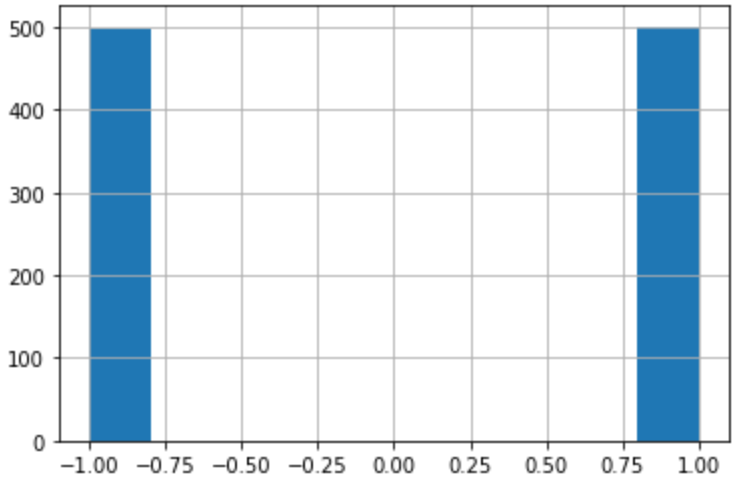


Figura 1: Distribuição das classes.

Ademais um pequeno adendo, para avaliar os classificadores nesta lista escolhemos o uso da acurácia, visto que queremos saber quão bem foram classificadas ambas as classes.

Item a)

Neste item é pedida a implementação de uma rede MLP com uma camada intermediária apenas. Como desejamos resolver um problema de classificação binária nossa camada de saída irá possuir um neurônio e escolhemos usar a função logística como ativação deste neurônio pois esta se adequa bem a problemas de classificação. A primeira coisa que notamos ao atacar esse problema é a dificuldade em estudar todos os parâmetros possíveis envolvendo o treinamento da rede. Para a estrutura da rede em si (independente do treinamento) teremos basicamente 2 decisões a tomar: a função de ativação da camada intermediária e o número de neurônios da camada intermediária. Como no item "d" iremos estudar a variação do número de neurônios da camada intermediária. Aproveitamos este item para estudar como duas funções de ativação diferentes para a camada intermediária se comportam mantendo fixo um número de neurônios igual a 40. Utilizaremos a ReLU e a logística. Além disso como a rede neural tem em sua saída uma função logística e pretendemos realizar uma classificação, utilizaremos como threshold o valor de 0,5(este é o valor padrão de threshold para a acurácia implementada no pacote Keras [1]). Também

verificamos que conforme a documentação do Keras as camadas densas implementadas apresentam bias [2].

Além disso devido a estrutura da camada de saída da rede neural (função logística com valores entre 0 e 1) alteramos os identificadores das classes de 1—>1 e de -1—>0. Explicitados estes detalhes acerca da rede que implementamos resta falar sobre o treinamento. Neste ponto reside a dificuldade pois temos inúmeros parâmetros:

- Otimizador: Adam, Gradiente Estocástico com momento de Nesterov (e seus parâmetros adicionais como taxa de aprendizado).
- Inicialização dos pesos.
- Valor de batch para cálculo do gradiente descendente.
- Função custo a ser utilizada (quadrados minimos, entropia cruzada).
- Número de épocas e critério de parada.

Além da função de ativação neste teste iremos também variar o método otimizador entre o Adam e o Sgd mantendo seus valores padrões. Os outros parâmetros serão setados conforme tabela abaixo:

Parâmetro	Valor
Inicialização dos Pesos	Default
Função Custo	Entropia Cruzada
Batch	100 (10% do total de dados de treino)
Numero de épocas	5000

Desta maneira prosseguimos com os nossos testes obtendo os resultados abaixo:

Teste	Função de Ativação	Otimizador	Tempo Treino	Custo Final Treino	Custo Final Validação	Acurácia Final Treino	Acurácia Final Validação
1	Relu	Adam	123	0,2585	0,3226	89,7%	86,70%
2	Relu	Sgd	122	0,2794	0,3186	88,6%	86,9%
3	Logística	Adam	123	0,6884	0,3149	86,7%	85,5%
4	Logística	Sgd	116	0,5524	0,5634	67%	65,3%

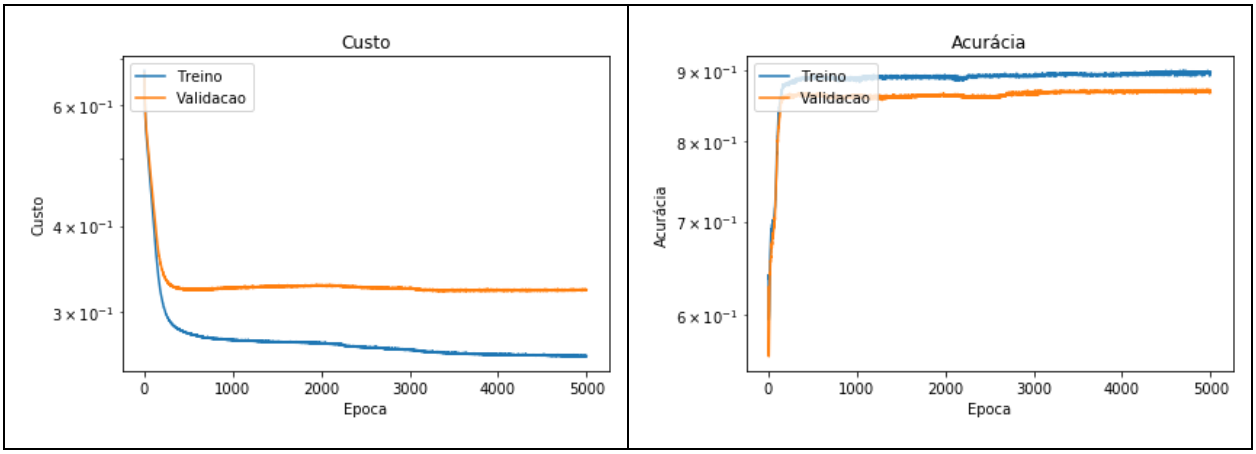


Figura 2: Teste 1, utilizando função ReLu e otimizador Adam.

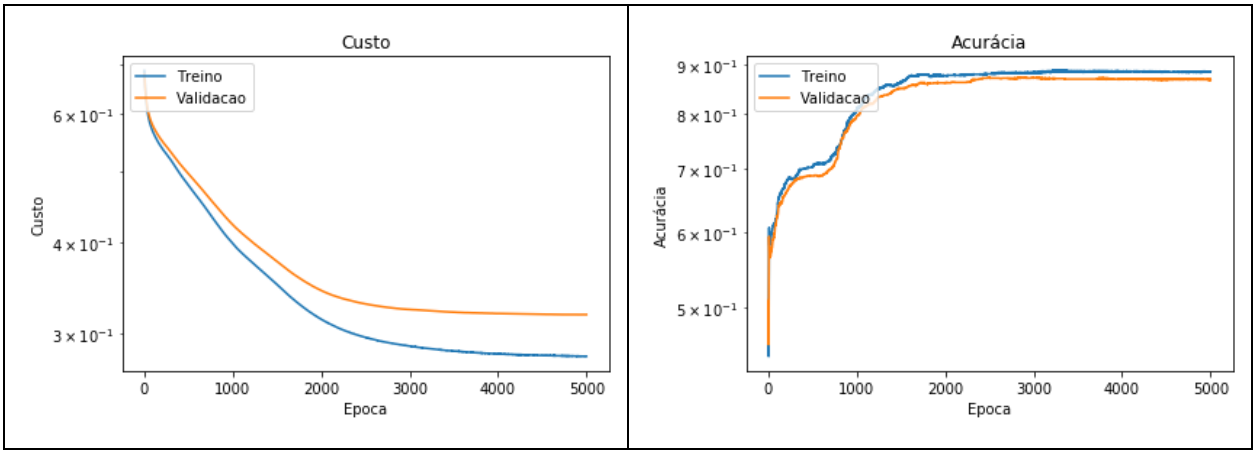


Figura 3: Teste 2, utilizando função ReLu e otimizador Sgd.

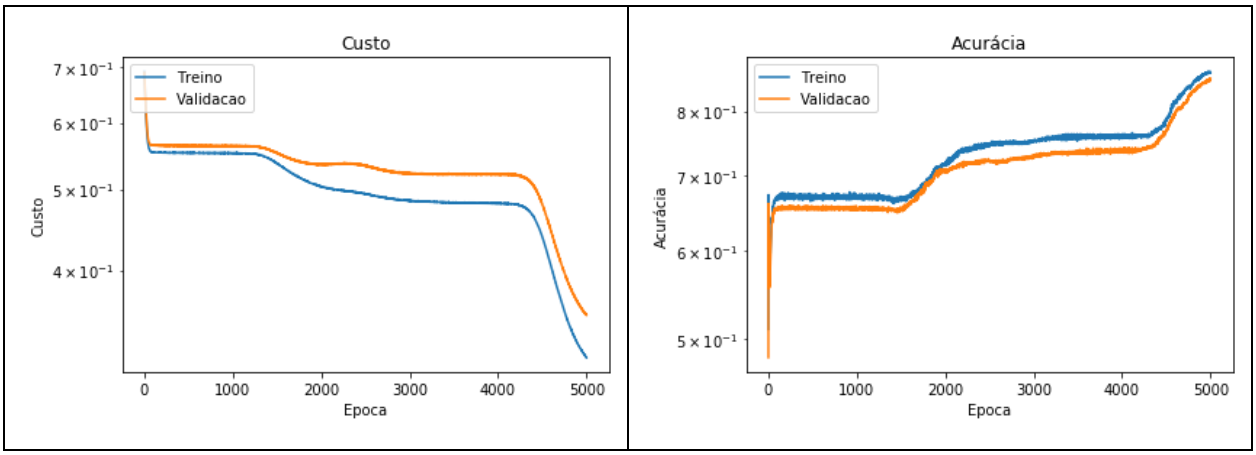


Figura 4: Teste 3, utilizando função logística e otimizador Adam.

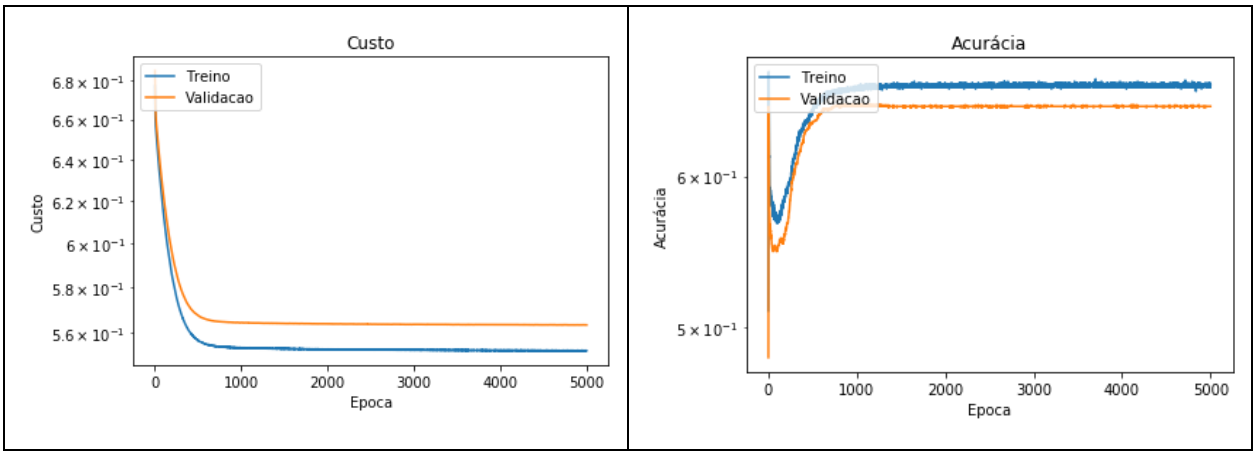


Figura 5: Teste 4, utilizando função logística e otimizador Sgd.

Comparando os métodos Adam e Sgd com a utilização da função Relu percebemos uma convergência mais direta do valor da acurácia com o Adam. Já para a função logística a convergência parece ser mais direta com o SGD. De qualquer forma a melhor acurácia de validação foi obtida com a função logística utilizando o método Sgd e portanto será a que iremos utilizar a partir deste ponto. Vale também notar a péssima acurácia obtida com o método sgd e utilizando a função logística, talvez devido à alguns mínimos locais (este teste foi feito duas vezes e portanto com uma geração de pesos aleatórios diferentes e mesmo assim os resultados foram similares). Podemos verificar nestes testes também que aparentemente não sofremos com overfitting em nenhum dos casos, pois a acurácia de validação sempre acompanha a de treino aumentando ao longo das épocas de treinamento.

Item b)

Para este item utilizamos a rede anterior com a função de ativação Relu e o otimizador Sgd. Como escolhemos o Sgd vale explicitar que por padrão (o que foi utilizado neste trabalho) ele implementa uma descida pro gradiente clássica com taxa de aprendizado de 0,01 e sem momento de Nesterov. Os outros parâmetros foram deixados idênticos aos anteriores. Desta forma geramos a seguinte fronteira de decisão:

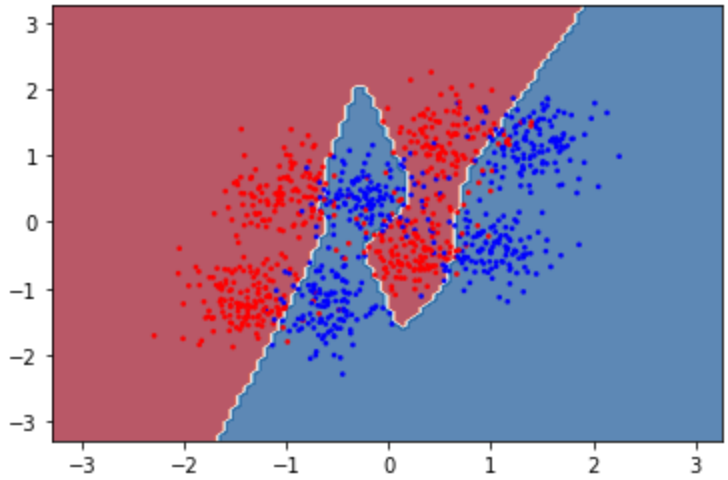


Figura 6: Fronteira de decisão gerada com a MLP de 40 neurônios e os pontos de treinamento utilizados .

Podemos notar que com essa fronteira de decisão temos problemas em classificar alguns dos pontos da área central do gráfico. Uma coisa interessante é que a fronteira de decisão parece ser meio escalonada como uma escada, talvez porque cada neurônio atue em uma parte da fronteira com sua função Relu que nada mais é do que uma reta para os valores positivos, porém tal idéia se esvai ao plotarmos a fronteira de decisão de uma rede com 5 neurônios por exemplo, pois notamos um grande número de “degraus” mesmo com esse número baixo de neurônios.

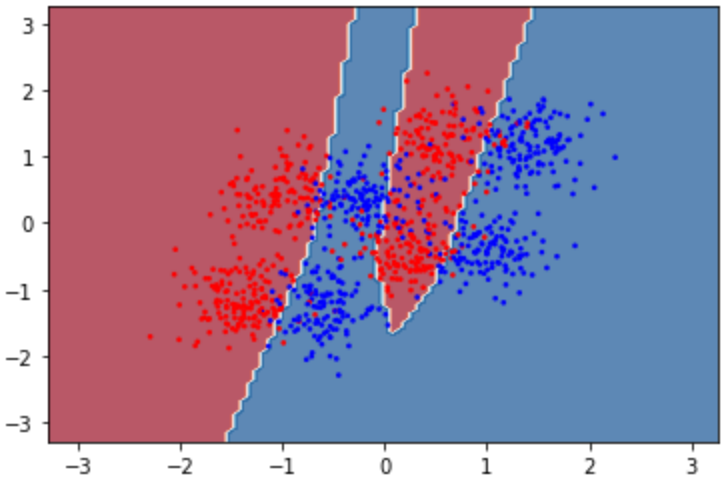


Figura 7: Fronteira de decisão gerada com a MLP de 5 neurônios e os pontos de treinamento utilizados .

Item c)

Sobre os dados de teste nossa rede teve uma acurácia de 88,8% e portanto um erro percentual de 11,2%. Aqui notamos que a acurácia apesar de menor do que a de treino como esperado foi ligeiramente maior do que a de validação o que é possível mas incomum.

Item d)

Aproveitamos este item para implementar um early stopping na rede neural com critério de variação da acurácia de validação de 0,0001% com paciência de 500 épocas (esta paciência teve que ser alta devido a alguns platos no gráfico da acurácia). Como modelo final foi utilizado o que atingiu a melhor acurácia de validação durante o treino e não o modelo final. Os resultados foram os seguintes:

Neuronios	Parada por Early Stopping	Epocas	Acurácia Treino	Acurácia Validação
1	Sim	1072	69,3	68,6
5	Sim	1979	86,1	85,1
10	Sim	3395	88,0	87,2
20	Sim	2252	87,9	85,9
40	Sim	2622	87,5	86,3
80	Sim	2107	87,3	86,5
160	Sim	3071	88,1	86,7
320	Sim	2155	87,7	86,5
640	Sim	2229	87,8	86,5

A rede com 10 neurônios já parece ser um bom aproximador com acurácia de validação de 87,2%, aumentando o número de neurônios não foi possível aumentar a acurácia talvez devido a limitações nos dados (número de dados insuficientes ou pouco representativos dos dados de teste).

Item e)

Para iniciar o projeto de um classificador utilizando SVM, foi escolhido o kernel gaussiano (rbf), por ser mais utilizado [9]. O requisito tepara usar esse tipo de classificador é que os dados sejam distribuídos seguindo um Gaussiana de média zero e variância 1 [13]. O formato do histograma abaixo com os dados de treinos e de validação mostra uma aproximação da distribuição com a distribuição normal.

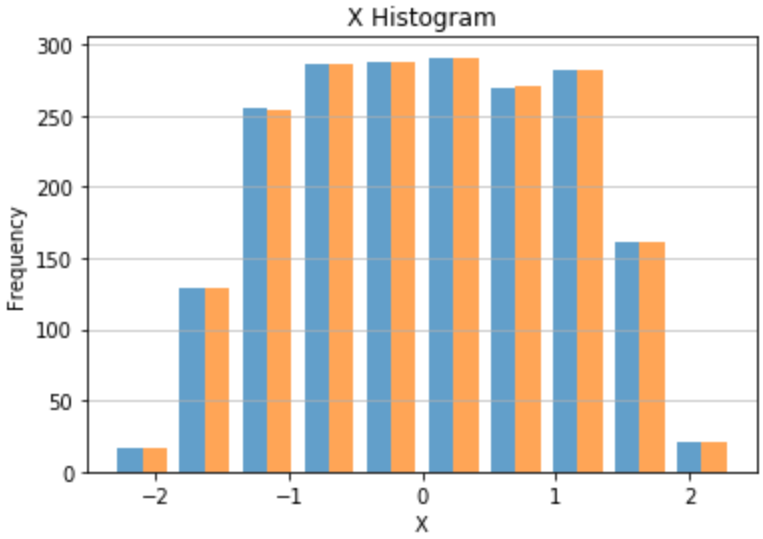


Figura: Histograma com os dados de treino e validação.
Formato se aproxima de uma distribuição normal.

Logo, como o SVM teóricamente necessita de dados standarlizados, apesar dos dados já estarem aparentemente standarlizados (com média dos atributos de treinamento (x1, x2) igual a [-0.01650581 -0.01679264] por exemplo), foi aplicada a standarlização dos dados. Além disso, a técnica também foi aplicada nos dados de validação e teste. Entretanto, foram utilizados os parâmetros, média e variância, dos dados de treinamento.

Há dois hiperparâmetro em questão para serem explorados no modelo do classificador: o gamma e o coeficiente de penalização, C. Para encontrá-los, foi realizada uma busca em grade com gamma e C variando entre [0,001 e 1000] com razão geométrica igual a 10, seguindo o diagrama de atividade em UML apresentado abaixo.

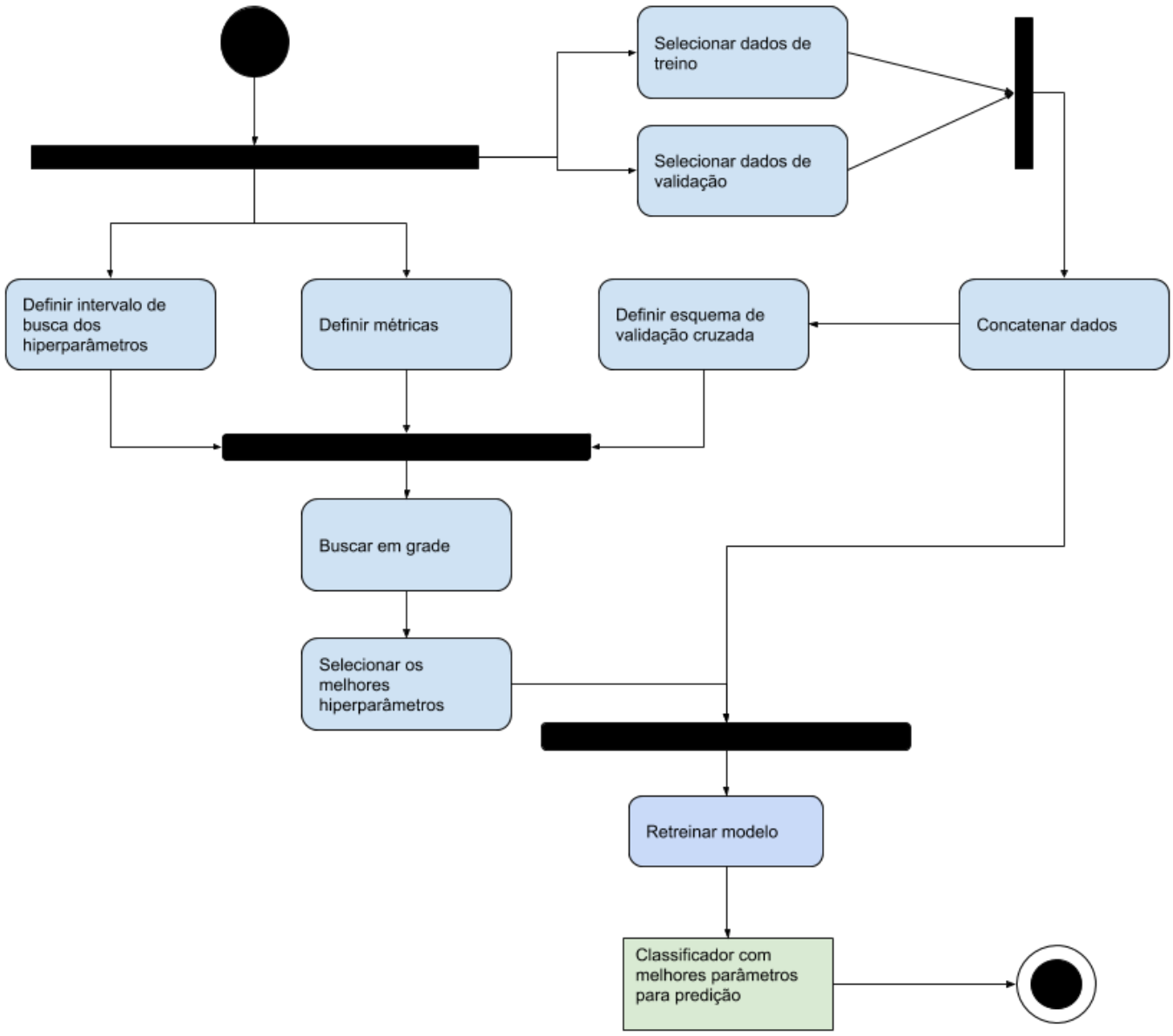


Figura: Diagrama de atividades para encontrar o melhor classificador usando busca em grade.

Foi utilizado um esquema de validação cruzada K-fold igual a 2 com os dados de treino com os dados de validação concatenados. As métricas utilizadas para verificar a performance do classificador foram a precisão (*precision*), sensibilidade (*recall*) e acurácia (*accuracy)*. Para verificar a generalização do modelo treinado, foram verificadas as métricas na fase de validação, obtendo precisão, recall e acurácia de aproximadamente 88%. O melhor valor dos parâmetros encontrado foi para gamma igual a 10 e C igual a 0.1 com todas as métricas. Com isso, o classificador foi treinado novamente utilizando os dados concatenados.

O gráfico abaixo apresenta a superfície de decisão das classes: vermelha para classe positiva e azul para a classe negativa. Foram plotados também os vetores suportes, que representam os pontos azul mais claro.

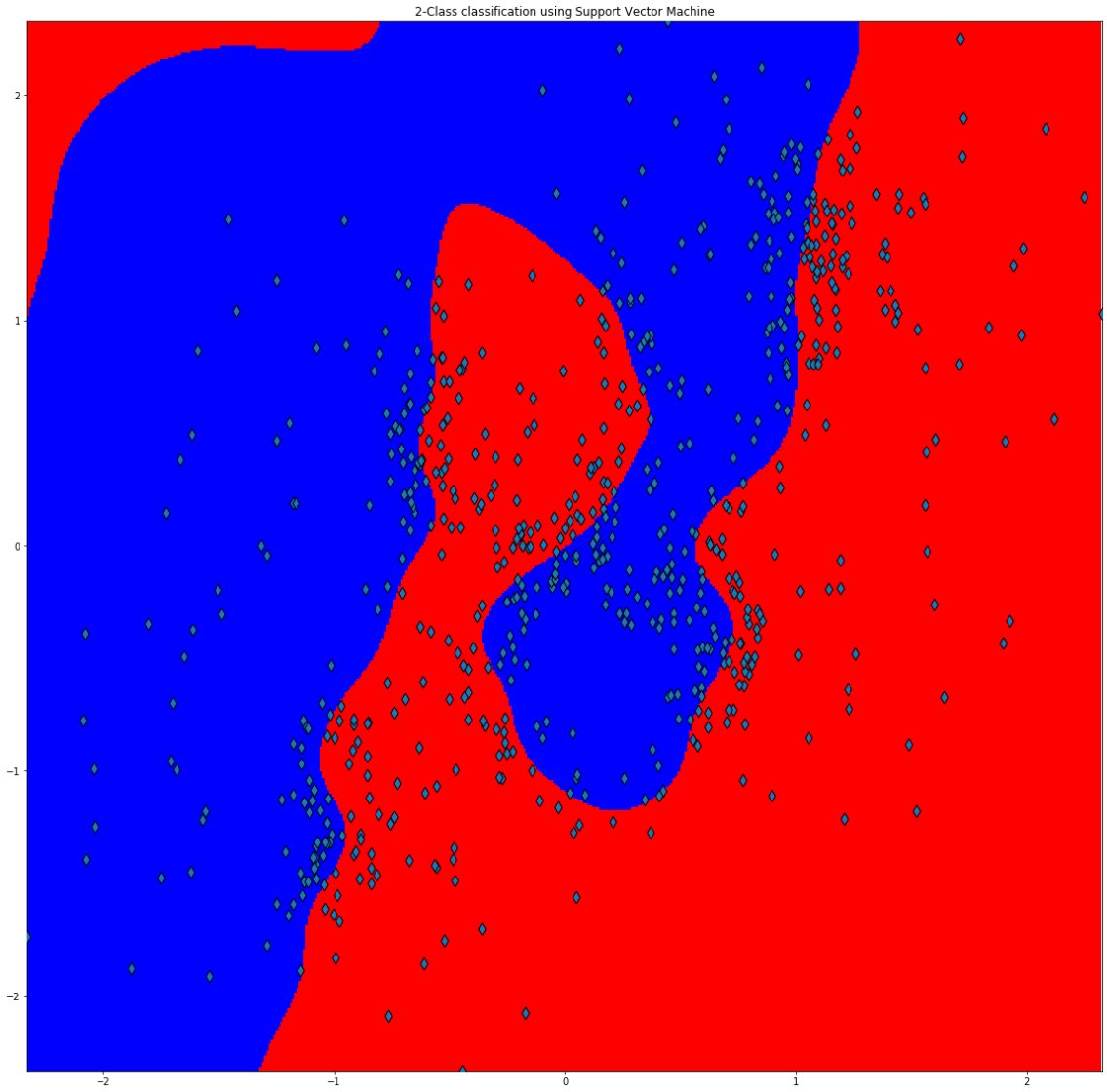


Figura: Região de decisão do classificador binário usando SVM com kernel rbf: vermelha para classe positiva e azul para a classe negativa.

Item f)

Aplicando a SVM treinada sobre o conjunto de teste com 1000 amostras, obteve-se a acurácia aproximada de 88%, o que implica em uma taxa de erro de 12%. Observa-se, portanto, que o classificador se comportou de maneira semelhante ao comparar com as métricas encontradas na fase validação (item e).

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	*recall*	*f1-score*	*support*
-1.0	*0.88*	*0.88*	*0.88*	*499*
1.0	*0.88*	*0.88*	*0.88*	*501*
accuracy			*0.88*	*1000*
macro avg	*0.88*	*0.88*	*0.88*	*1000*
weighted avg	*0.88*	*0.88*	*0.88*	*1000*

Number of support vector: [345 346]

Item g)

Variando o parâmetro gamma do kernel RBF

Sabe-se que o parâmetro gamma define quão longe uma amostra dos dados de treino influência no espaço de atributos, ou seja, quanto menor, mais longe a amostra influência pelo espaço. Por outro lado, enquanto maior, menor é a influência nas regiões mais próximas.

Inicialmente, observou-se a distribuição dos dados de treino concatenados com os dados de validação no espaço dos atributos (x1 e x2). A figura abaixo mostra essa distribuição bem como a reta x1 = x2. Na parte debaixo da reta, há maior concentração de pontos vermelhos. Já na parte de cima, há uma maior concentração de pontos azuis.

Portanto, é de se esperar que com gamma pequeno, área em cima da reta seja influenciada mais pelos pontos azuis do que os vemelhos. Já na área embaixo, pelos pontos vermelhos. As figuras abaixo apresentam as regiões de decisão que demonstra esse comportamento para diferentes valores de gamma. Foi utilizado um mapa de cores divergente *bwr* (blue-white-red) para facilitar a visualização. A região branca representa a fronteira de decisão.

É fácil de verificar pela regiões que, a medida que o gamma, menor a região que cada ponto influencia. Além disso, a região que representa a fronteira de decisão é mais complexa a medida que o gamma aumenta. No caso limite, a fronteira se colapsa nos próprios dados de treino.

As métricas referente ao dados de testes foram explicitadas nos quadros abaixo, sendo a menor com gamma igual a 0.001 e a maior com gamma igual a 1.

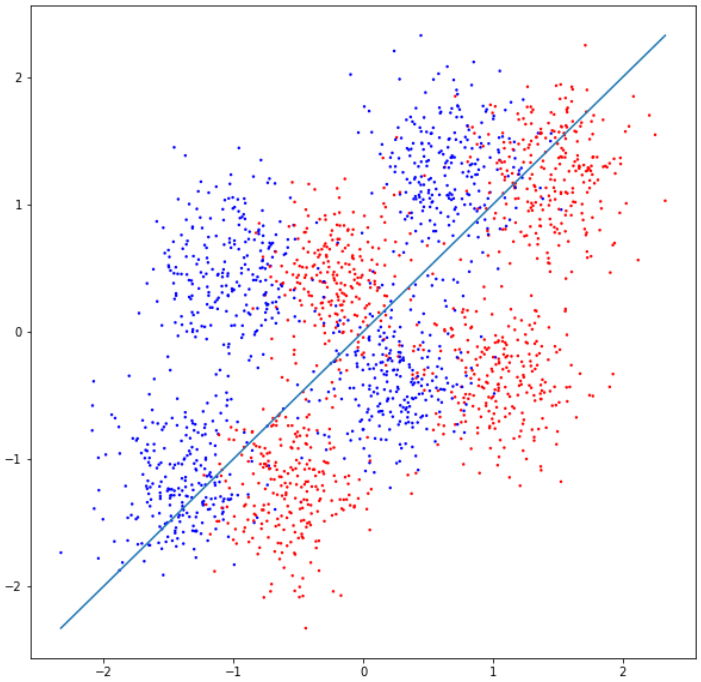


Figura: Conjunto de dados no espaço dos atributos.
Cada cor representa uma classe: vermelha C+, azul C-.

Região de decisão: gamma = 0.001 e C = 1

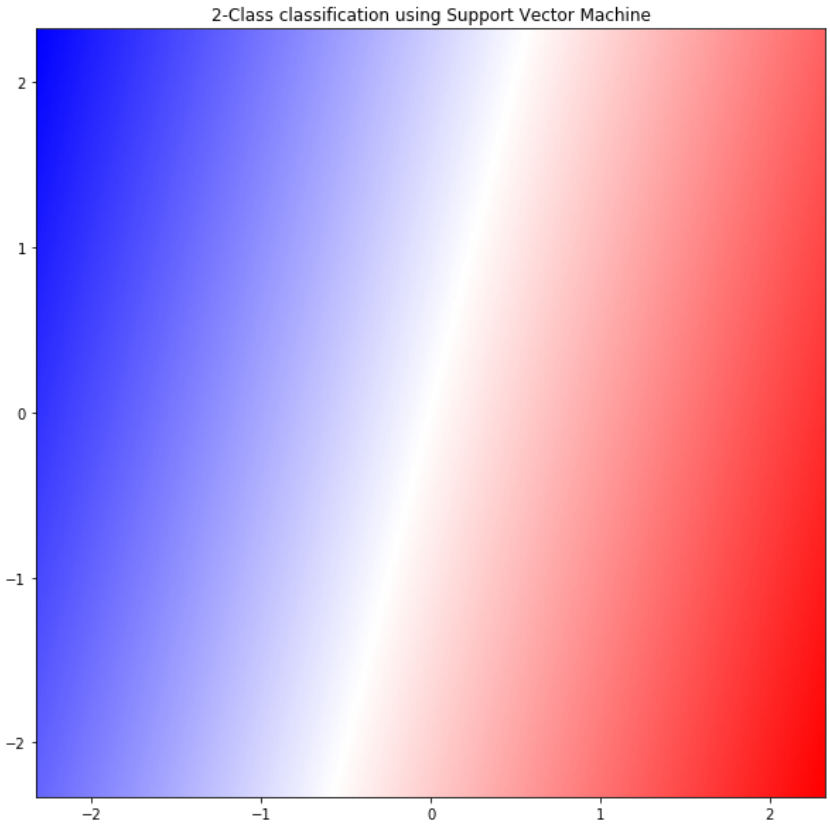


Figura: Região de decisão para gamma igual a 0.001 e C = 1

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	*recall*	*f1-score*	*support*
-1.0	*0.58*	*0.56*	*0.57*	*499*
1.0	*0.58*	*0.59*	*0.58*	*501*
accuracy			*0.58*	*1000*
macro avg	*0.58*	*0.58*	*0.58*	*1000*
weighted avg	*0.58*	*0.58*	*0.58*	*1000*

Região de decisão: gamma = 1 e C = 1

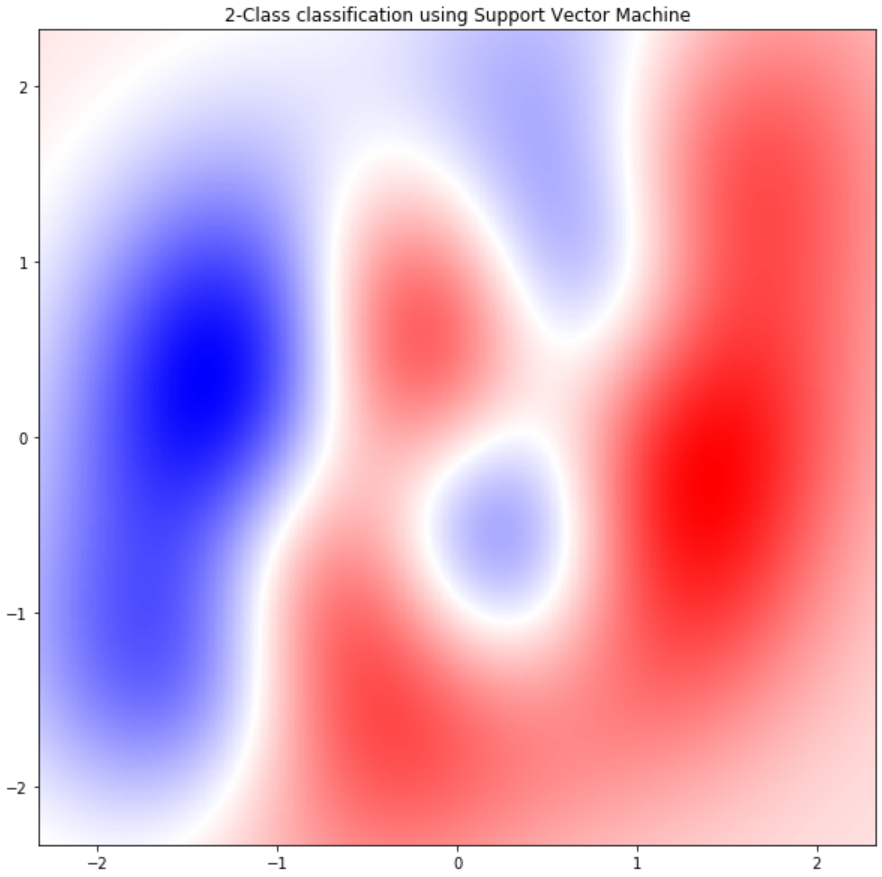


Figura: Região de decisão para gamma igual a 1 e C igual a 1

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	recall	f1-score	support
-1.0	0.88	0.89	0.88	499
1.0	0.89	0.88	0.88	501
accuracy			0.88	1000
macro avg	0.88	0.88	0.88	1000
weighted avg	0.88	0.88	0.88	1000

Região de decisão: gamma = 1000 e C = 1

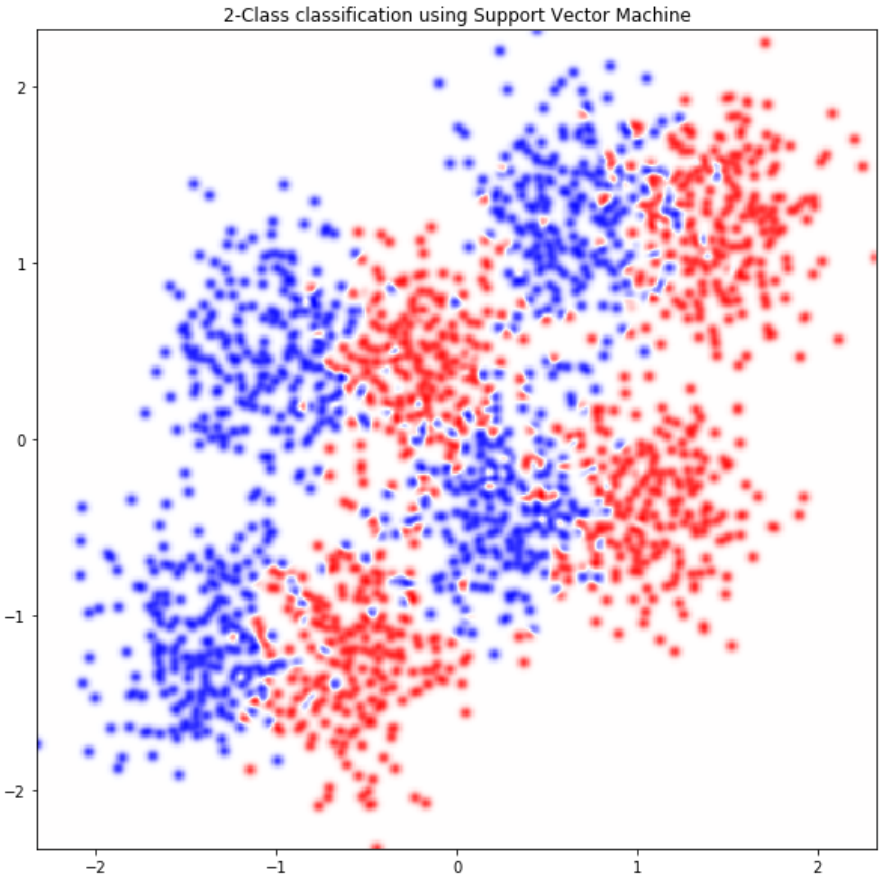


Figura: Região de decisão para gamma igual a 1000 e C igual a 1

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	*recall*	*f1-score*	*support*
-1.0	*0.83*	*0.73*	*0.77*	*499*
1.0	*0.76*	*0.85*	*0.80*	*501*
accuracy			*0.79*	*1000*
macro avg	*0.79*	*0.79*	*0.79*	*1000*
weighted avg	*0.79*	*0.79*	*0.79*	*1000*

Variando o parâmetro de penalização do kernel RBF

Sabe-se que a variação termo de penalização leva a um trade off entre classificar as amostras corretamente e maximizar a margem. Para valores maiores de C, uma margem menor será aceita se a função de decisão for melhor na classificação correta de todos os pontos de treinamento. Um C mais baixo incentivará uma margem maior, portanto, uma função de decisão mais simples, ao custo da precisão do treinamento. Em outras palavras, C se comporta como um parâmetro de regularização no SVM [12].

As figuras abaixo apresentam as regiões de decisão que demonstra esse comportamento para diferentes valores de C. Foi utilizado um mapa de cores divergente *bwr* (blue-white-red) para facilitar a visualização. A região branca representa a fronteira de decisão.

Então, é possível verificar que a região de decisão para C igual a 0.001 é mais simples comparado com os outros valores ao custo da precisão. Ao analisar as métricas com relação aos dados de teste, é possível constatar que precisão está em torno de 50% e aumenta na medida que o valor de C aumenta, chegando em 89%.

Região de decisão: gamma = 1 e C = 0.001

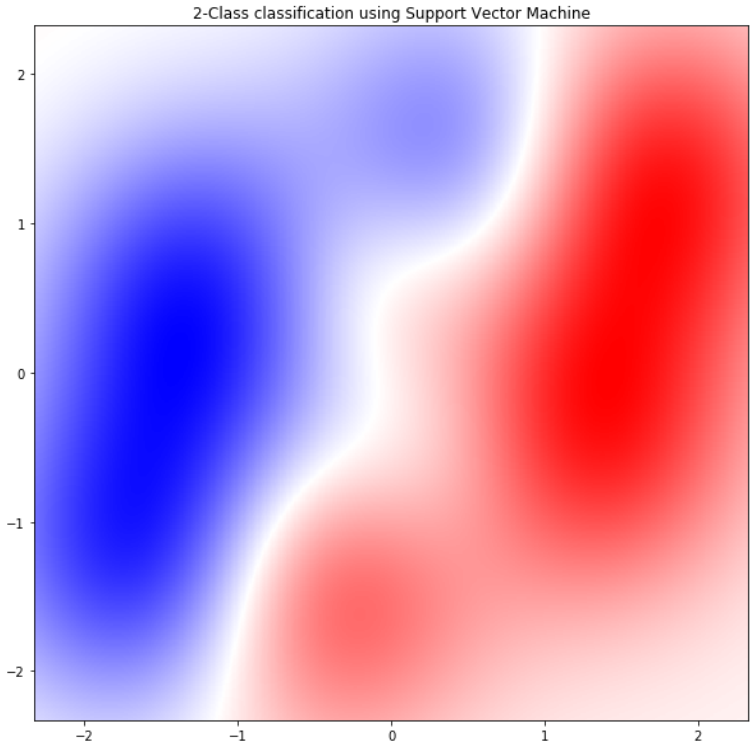


Figura: Região de decisão para gamma igual a 1 e C igual a 0.001

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	*recall*	*f1-score*	*support*
-1.0	*0.00*	*0.00*	*0.00*	*499*
1.0	*0.50*	*1.00*	*0.67*	*501*
accuracy			*0.50*	*1000*
macro avg	*0.25*	*0.50*	*0.33*	*1000*
weighted avg	*0.25*	*0.50*	*0.33*	*1000*

Região de decisão: gamma = 1 e C = 1

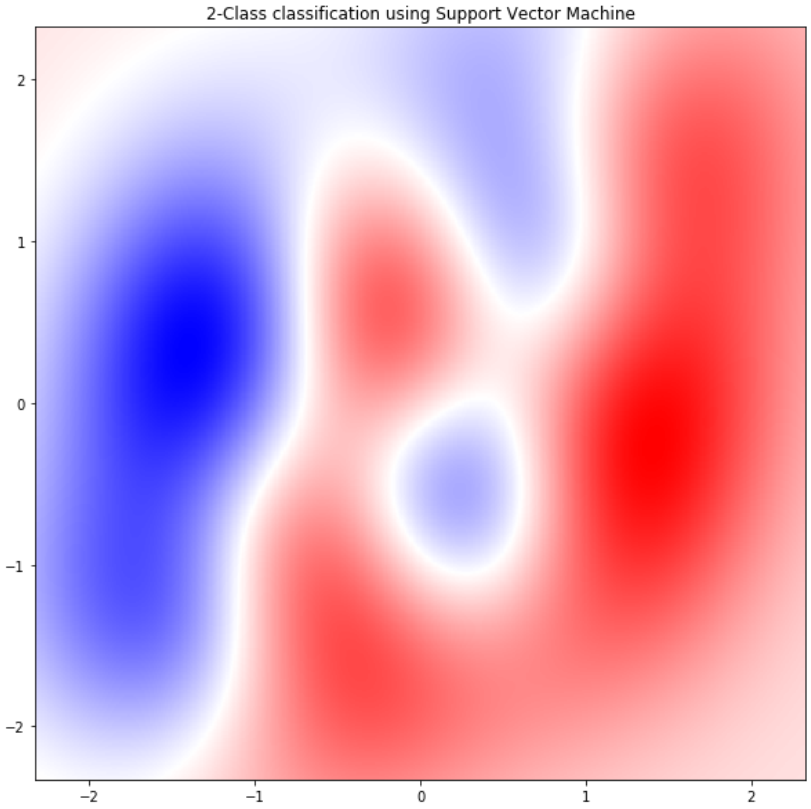


Figura: Região de decisão para gamma igual a 1 e C igual a 1

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	*recall*	*f1-score*	*support*
-1.0	*0.88*	*0.89*	*0.88*	*499*
1.0	*0.89*	*0.88*	*0.88*	*501*
accuracy			*0.88*	*1000*
macro avg	*0.88*	*0.88*	*0.88*	*1000*
weighted avg	*0.88*	*0.88*	*0.88*	*1000*

Região de decisão: gamma = 1 e C = 1000

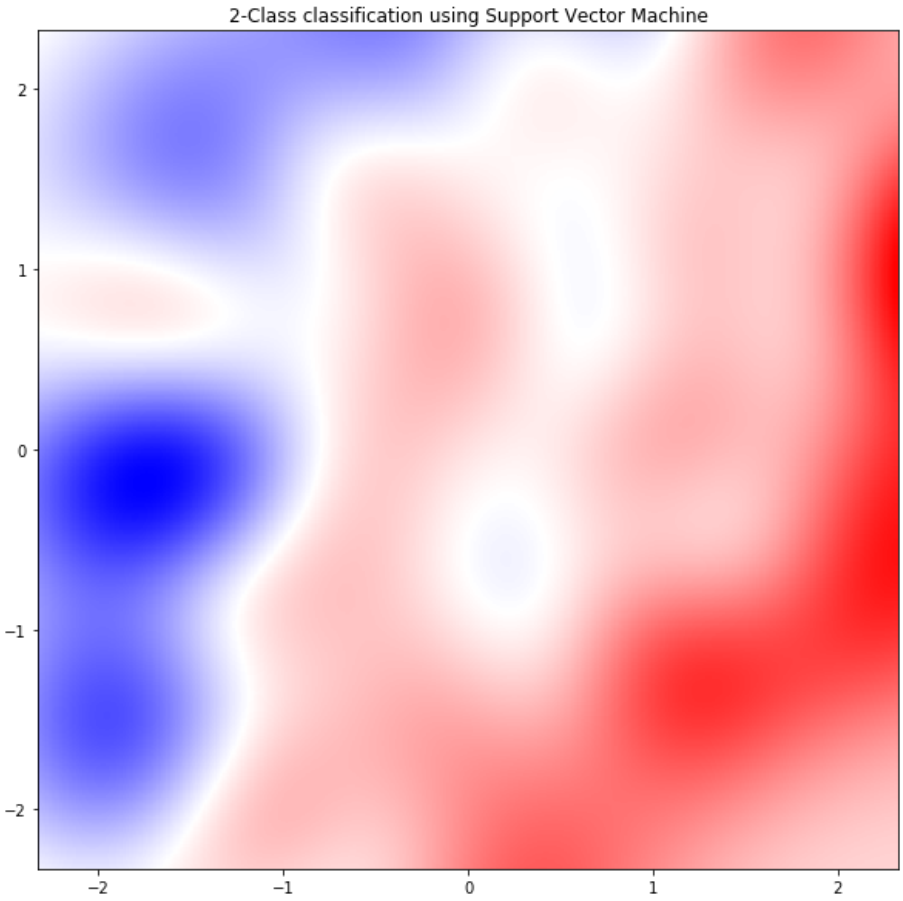


Figura: Região de decisão para gamma igual a 1 e C igual a 1000

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	*recall*	*f1-score*	*support*
-1.0	*0.89*	*0.87*	*0.88*	*499*
1.0	*0.87*	*0.89*	*0.88*	*501*
accuracy			*0.88*	*1000*
macro avg	*0.88*	*0.88*	*0.88*	*1000*
weighted avg	*0.88*	*0.88*	*0.88*	*1000*

ANEXO A - Tuning hyper-parameters for precision

Tuning hyper-parameters for precision

Best parameters set found on development set:

{'C': 1.0, 'gamma': 10.0}

Grid scores on development set:

0.504 (+/-0.000) for {'C': 0.001, 'gamma': 0.001}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 0.01}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 0.1}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 1.0}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 10.0}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 100.0}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 1000.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 0.001}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 0.01}
0.673 (+/-0.085) for {'C': 0.01, 'gamma': 0.1}
0.713 (+/-0.012) for {'C': 0.01, 'gamma': 1.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 10.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 100.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 1000.0}
0.504 (+/-0.000) for {'C': 0.1, 'gamma': 0.001}
0.564 (+/-0.023) for {'C': 0.1, 'gamma': 0.01}
0.619 (+/-0.055) for {'C': 0.1, 'gamma': 0.1}
0.860 (+/-0.009) for {'C': 0.1, 'gamma': 1.0}
0.875 (+/-0.002) for {'C': 0.1, 'gamma': 10.0}
0.599 (+/-0.054) for {'C': 0.1, 'gamma': 100.0}
0.504 (+/-0.000) for {'C': 0.1, 'gamma': 1000.0}
0.566 (+/-0.018) for {'C': 1.0, 'gamma': 0.001}
0.625 (+/-0.040) for {'C': 1.0, 'gamma': 0.01}
0.710 (+/-0.023) for {'C': 1.0, 'gamma': 0.1}
0.874 (+/-0.001) for {'C': 1.0, 'gamma': 1.0}
0.877 (+/-0.005) for {'C': 1.0, 'gamma': 10.0}
0.856 (+/-0.001) for {'C': 1.0, 'gamma': 100.0}
0.728 (+/-0.018) for {'C': 1.0, 'gamma': 1000.0}

0.629 (+/-0.033) for {'C': 10.0, 'gamma': 0.001}
0.669 (+/-0.048) for {'C': 10.0, 'gamma': 0.01}
0.744 (+/-0.018) for {'C': 10.0, 'gamma': 0.1}
0.873 (+/-0.002) for {'C': 10.0, 'gamma': 1.0}
0.873 (+/-0.015) for {'C': 10.0, 'gamma': 10.0}
0.822 (+/-0.024) for {'C': 10.0, 'gamma': 100.0}
0.727 (+/-0.008) for {'C': 10.0, 'gamma': 1000.0}
0.670 (+/-0.042) for {'C': 100.0, 'gamma': 0.001}
0.686 (+/-0.036) for {'C': 100.0, 'gamma': 0.01}
0.840 (+/-0.008) for {'C': 100.0, 'gamma': 0.1}
0.875 (+/-0.003) for {'C': 100.0, 'gamma': 1.0}
0.866 (+/-0.011) for {'C': 100.0, 'gamma': 10.0}
0.799 (+/-0.004) for {'C': 100.0, 'gamma': 100.0}
0.735 (+/-0.029) for {'C': 100.0, 'gamma': 1000.0}
0.674 (+/-0.039) for {'C': 1000.0, 'gamma': 0.001}
0.724 (+/-0.017) for {'C': 1000.0, 'gamma': 0.01}
0.864 (+/-0.010) for {'C': 1000.0, 'gamma': 0.1}
0.875 (+/-0.003) for {'C': 1000.0, 'gamma': 1.0}
0.845 (+/-0.019) for {'C': 1000.0, 'gamma': 10.0}
0.795 (+/-0.012) for {'C': 1000.0, 'gamma': 100.0}
0.735 (+/-0.029) for {'C': 1000.0, 'gamma': 1000.0}

Refit with all data (train + validation):

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	recall	f1-score	support
-1.0	0.88	0.88	0.88	499
1.0	0.88	0.88	0.88	501
accuracy			0.88	1000
macro avg	0.88	0.88	0.88	1000
weighted avg	0.88	0.88	0.88	1000

Number of support vector: [345 346]

ANEXO B - Tuning hyper-parameters for recall

Tuning hyper-parameters for recall

Best parameters set found on development set:

{'C': 1.0, 'gamma': 10.0}

Grid scores on development set:

0.504 (+/-0.000) for {'C': 0.001, 'gamma': 0.001}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 0.01}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 0.1}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 1.0}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 10.0}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 100.0}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 1000.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 0.001}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 0.01}
0.673 (+/-0.085) for {'C': 0.01, 'gamma': 0.1}
0.713 (+/-0.012) for {'C': 0.01, 'gamma': 1.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 10.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 100.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 1000.0}
0.504 (+/-0.000) for {'C': 0.1, 'gamma': 0.001}
0.564 (+/-0.023) for {'C': 0.1, 'gamma': 0.01}
0.619 (+/-0.055) for {'C': 0.1, 'gamma': 0.1}
0.860 (+/-0.009) for {'C': 0.1, 'gamma': 1.0}
0.875 (+/-0.002) for {'C': 0.1, 'gamma': 10.0}
0.599 (+/-0.054) for {'C': 0.1, 'gamma': 100.0}
0.504 (+/-0.000) for {'C': 0.1, 'gamma': 1000.0}
0.566 (+/-0.018) for {'C': 1.0, 'gamma': 0.001}
0.625 (+/-0.040) for {'C': 1.0, 'gamma': 0.01}
0.710 (+/-0.023) for {'C': 1.0, 'gamma': 0.1}
0.874 (+/-0.001) for {'C': 1.0, 'gamma': 1.0}
0.877 (+/-0.005) for {'C': 1.0, 'gamma': 10.0}
0.856 (+/-0.001) for {'C': 1.0, 'gamma': 100.0}
0.728 (+/-0.018) for {'C': 1.0, 'gamma': 1000.0}
0.629 (+/-0.033) for {'C': 10.0, 'gamma': 0.001}
0.669 (+/-0.048) for {'C': 10.0, 'gamma': 0.01}
0.744 (+/-0.018) for {'C': 10.0, 'gamma': 0.1}

0.873 (+/-0.002) for {'C': 10.0, 'gamma': 1.0}
0.873 (+/-0.015) for {'C': 10.0, 'gamma': 10.0}
0.822 (+/-0.024) for {'C': 10.0, 'gamma': 100.0}
0.727 (+/-0.008) for {'C': 10.0, 'gamma': 1000.0}
0.670 (+/-0.042) for {'C': 100.0, 'gamma': 0.001}
0.686 (+/-0.036) for {'C': 100.0, 'gamma': 0.01}
0.840 (+/-0.008) for {'C': 100.0, 'gamma': 0.1}
0.875 (+/-0.003) for {'C': 100.0, 'gamma': 1.0}
0.866 (+/-0.011) for {'C': 100.0, 'gamma': 10.0}
0.799 (+/-0.004) for {'C': 100.0, 'gamma': 100.0}
0.735 (+/-0.029) for {'C': 100.0, 'gamma': 1000.0}
0.674 (+/-0.039) for {'C': 1000.0, 'gamma': 0.001}
0.724 (+/-0.017) for {'C': 1000.0, 'gamma': 0.01}
0.864 (+/-0.010) for {'C': 1000.0, 'gamma': 0.1}
0.875 (+/-0.003) for {'C': 1000.0, 'gamma': 1.0}
0.845 (+/-0.019) for {'C': 1000.0, 'gamma': 10.0}
0.795 (+/-0.012) for {'C': 1000.0, 'gamma': 100.0}
0.735 (+/-0.029) for {'C': 1000.0, 'gamma': 1000.0}

Refit with all data (train + validation):

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	recall	f1-score	support
-1.0	0.88	0.88	0.88	499
1.0	0.88	0.88	0.88	501
accuracy			0.88	1000
macro avg	0.88	0.88	0.88	1000
weighted avg	0.88	0.88	0.88	1000

Number of support vector: [345 346]

ANEXO C - Tuning hyper-parameters for accuracy

Tuning hyper-parameters for accuracy

Best parameters set found on development set:

{'C': 1.0, 'gamma': 10.0}

Grid scores on development set:

0.504 (+/-0.000) for {'C': 0.001, 'gamma': 0.001}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 0.01}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 0.1}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 1.0}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 10.0}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 100.0}
0.504 (+/-0.000) for {'C': 0.001, 'gamma': 1000.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 0.001}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 0.01}
0.673 (+/-0.085) for {'C': 0.01, 'gamma': 0.1}
0.713 (+/-0.012) for {'C': 0.01, 'gamma': 1.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 10.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 100.0}
0.504 (+/-0.000) for {'C': 0.01, 'gamma': 1000.0}
0.504 (+/-0.000) for {'C': 0.1, 'gamma': 0.001}
0.564 (+/-0.023) for {'C': 0.1, 'gamma': 0.01}
0.619 (+/-0.055) for {'C': 0.1, 'gamma': 0.1}
0.860 (+/-0.009) for {'C': 0.1, 'gamma': 1.0}
0.875 (+/-0.002) for {'C': 0.1, 'gamma': 10.0}
0.599 (+/-0.054) for {'C': 0.1, 'gamma': 100.0}
0.504 (+/-0.000) for {'C': 0.1, 'gamma': 1000.0}
0.566 (+/-0.018) for {'C': 1.0, 'gamma': 0.001}
0.625 (+/-0.040) for {'C': 1.0, 'gamma': 0.01}
0.710 (+/-0.023) for {'C': 1.0, 'gamma': 0.1}
0.874 (+/-0.001) for {'C': 1.0, 'gamma': 1.0}
0.877 (+/-0.005) for {'C': 1.0, 'gamma': 10.0}
0.856 (+/-0.001) for {'C': 1.0, 'gamma': 100.0}
0.728 (+/-0.018) for {'C': 1.0, 'gamma': 1000.0}
0.629 (+/-0.033) for {'C': 10.0, 'gamma': 0.001}
0.669 (+/-0.048) for {'C': 10.0, 'gamma': 0.01}
0.744 (+/-0.018) for {'C': 10.0, 'gamma': 0.1}

0.873 (+/-0.002) for {'C': 10.0, 'gamma': 1.0}
0.873 (+/-0.015) for {'C': 10.0, 'gamma': 10.0}
0.822 (+/-0.024) for {'C': 10.0, 'gamma': 100.0}
0.727 (+/-0.008) for {'C': 10.0, 'gamma': 1000.0}
0.670 (+/-0.042) for {'C': 100.0, 'gamma': 0.001}
0.686 (+/-0.036) for {'C': 100.0, 'gamma': 0.01}
0.840 (+/-0.008) for {'C': 100.0, 'gamma': 0.1}
0.875 (+/-0.003) for {'C': 100.0, 'gamma': 1.0}
0.866 (+/-0.011) for {'C': 100.0, 'gamma': 10.0}
0.799 (+/-0.004) for {'C': 100.0, 'gamma': 100.0}
0.735 (+/-0.029) for {'C': 100.0, 'gamma': 1000.0}
0.674 (+/-0.039) for {'C': 1000.0, 'gamma': 0.001}
0.724 (+/-0.017) for {'C': 1000.0, 'gamma': 0.01}
0.864 (+/-0.010) for {'C': 1000.0, 'gamma': 0.1}
0.875 (+/-0.003) for {'C': 1000.0, 'gamma': 1.0}
0.845 (+/-0.019) for {'C': 1000.0, 'gamma': 10.0}
0.795 (+/-0.012) for {'C': 1000.0, 'gamma': 100.0}
0.735 (+/-0.029) for {'C': 1000.0, 'gamma': 1000.0}

Refit with all data (train + validation):

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	recall	f1-score	support
-1.0	0.88	0.88	0.88	499
1.0	0.88	0.88	0.88	501
accuracy			0.88	1000
macro avg	0.88	0.88	0.88	1000
weighted avg	0.88	0.88	0.88	1000

Number of support vector: [345 346]

Código-fonte

- https://github.com/navarrothiago/ia006/tree/master/IA006-EFC03

Referências

1. https://stackoverflow.com/questions/41651011/binary-accuracy-in-keras-metrices-whats-the-threshold-value-to-predicted-one
2. https://keras.io/layers/core/#dense
3. https://scikit-learn.org/stable/auto_examples/model_selection/plot_grid_search_digits.html
4. https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation
5. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
6. https://stackoverflow.com/questions/48390601/explicitly-specifying-test-train-sets-in-gridsearchcv
7. https://stats.stackexchange.com/questions/321559/pre-processing-applied-on-all-three-training-validation-test-sets
8. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV
9. http://www.dca.fee.unicamp.br/~lboccato/topico_7.3_SVM.pdf
10. https://scikit-learn.org/stable/modules/grid_search.html#grid-search
11. https://scikit-learn.org/stable/auto_examples/svm/plot_custom_kernel.html#sphx-glr-auto-examples-svm-plot-custom-kernel-py
12. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html
13. https://scikit-learn.org/stable/modules/preprocessing.html
14. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC