Questão 1

A abordagem para esta questão foi feita por *Multilayer Perceptron* (MLP), e para isso o framework utilizado foi o Keras que usa como backend o tensorflow. O problema abordado foi a classificação na base de dados IRIS.

A base de dados escolhida consiste em um documento no formato csv com quatro características da flor, são elas: largura e comprimento da pétala e largura e comprimento da sépala além de um rótulo indicando a qual espécie ela pertence. É um conjunto de dados relativamente pequeno, com 150 amostras no total, 50 de cada classe.

É um problema muticlasse, pois há a possibilidade de uma flor ser classificada em 3 classes: setosa, versicolor e Virginia. Por se tratar de uma abordagem de mais de uma classe, a última camada da rede MLP teve sua ativação como *softmax* para os três neurônios na camada de saída.

Foi utilizado crossvaligdation, ou seja o conjunto de dados foi dividido em três partes: treino, validação e teste. Cada grupo ficou com uma porcentagem dos dados. O conjunto de treino ficou com a maior parte dos dados 80% para tentar fazer com que o modelo ficasse com a maior quantidade de dados para ser vista em seu treinamento e auxiliar na generalização dos casos. O conjunto de validação ficou com 10% das amostras do conjunto de dados e o conjunto de teste ficou com os outros 10%.

O conjunto de dados possui 4 características a serem analisadas, o que fez com que a rede neural ficasse com uma dimensão igual a 4. O modelo foi feito inicialmente com apenas uma camada densa com 64 neurônios e sua função de ativação sigmoid, em seguida uma camada de *dropout* foi adicionada com o argumento de 0.5 para que diminuísse a chance de ocorrer *overfitting* e a camada de saída com 3 neurônios (um para cada classe) com a função de ativação sendo a *softmax*.

Como a abordagem é mutilasse, houve a necessidade de fazer a transformação dos rótulos para *oneHotEncoding* o que os torna um vetor [1,0,0], [0,1,0] ou [0,0,1] para indicar as classes. Inicialmente o modelo foi compilado com o otimizador SGD com o *learning rate* de 0.1, a *loss* sendo *categorical_crossentropy* e como métrica a ser vista pelo modelo foi definida a acurácia.

Dois *callbacks* foram utilizados para auxiliar o treinamento do modelo, o *model_checkpoint* e o *early_stopping* o primeiro foi utilizado para salvar a melhor versão do modelo, que tivesse a menor *loss* que ocorresse durante o treinamento o segundo teve o intuito de verificar se o modelo deveria interromper o treinamento baseado na melhora da *val_loss* caso não fosse melhorada por 10 épocas o treinamento seria interrompido. A configuração inicial foi de 300 épocas, porém foi algo contando que o critério de parada adiantada aconteceria antes que as épocas configuradas acontecessem.

Após o treinamento o modelo foi avaliado e testado, e suas métricas foram expostas.

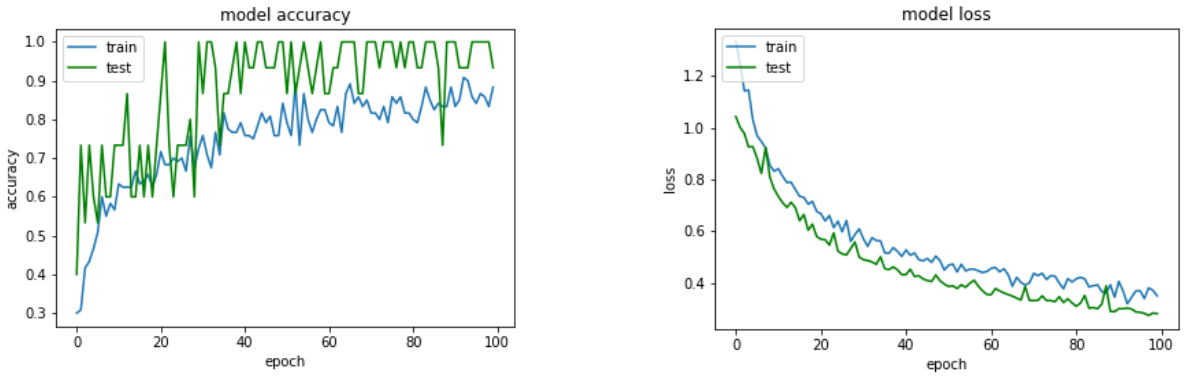


Figura1 - teste do modelo para 100 épocas sem early stopping

As imagens acima são da acurácia e da loss do modelo treinado para a primeira tentativa de treinamento sem a parada antecipada. É possível ver que os resultados estão próximos, porém algo inusitado é que o que normalmente é observado é o conjunto de teste ter uma performance melhor que o conjunto de validação ou teste, porém o contrário aconteceu neste caso. Provavelmente a pouca quantidade de amostras no conjunto de teste e validação pode ter

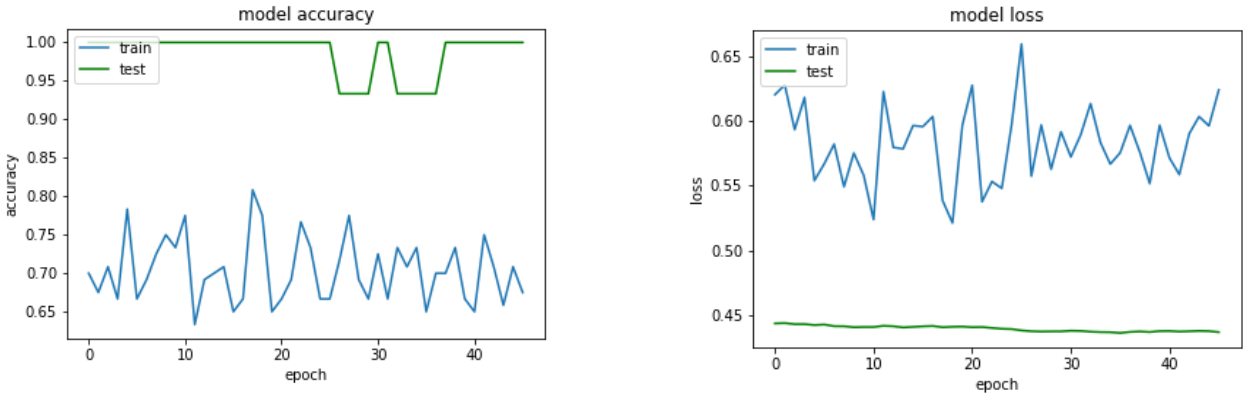


Figura2 - resultado do treino com a parada antecipada

acarretado tal resultado, dado que 10% do conjunto total 150 amostras resulta em apenas 15 amostras para cada conjunto (teste e validação). O treino com a parada antecipada é mostrado na figura 2.

Após o primeiro treinamento com a configuração mencionada a cima foi feito uma segunda abordagem para investigar os resultados, mais camadas foram adicionada à rede e então o segundo teste teve uma mudança no Learning rate diminuiu para 0.01 o critério de parada foi mantido.

O treinamento foi realizado com Jupyter notebook e possui mais detalhes do treinamento, o código está no GitHub assim como as questões subsequentes. O link de acesso ao repositório no GitHub é:https://github.com/Geovannioj/ML_secondList. O repositório contém uma pasta para cada questão da lista 2.
Link para o código específico da questão 1: https://github.com/Geovannioj/ML_secondList/tree/master/Questao1-MLP

Questão 2

O conjunto de dados utilizado foi o mesmo da questão anterior o Iris. O algoritmo utilizado foi o **J48** conforme solicitado na questão, os dados sofreram dois experimentos. O primeiro foi dividir o conjunto de dados ao meio 50% para teste e 50 para treino, e o segundo foi na proporção de 80% das amostras para treinamento e 20% para teste.
A acurácia para esse experimento foi condizente com a questão 1 que obteve 93.3%, assim como o valor obtido de 94% na WEKA, a matrix de confusão do primeiro experimento é mostrado na figura3 e a árvore de decisão construída com os dados é mostrado na figura 4.

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 22  0  0 |  a = setosa
  0 26  0 |  b = versicolor
  0  4 23 |  c = virginica
```

Figura3 - matrix de confusão do experimento com divisão dos dados em 50% treino e 50% teste.

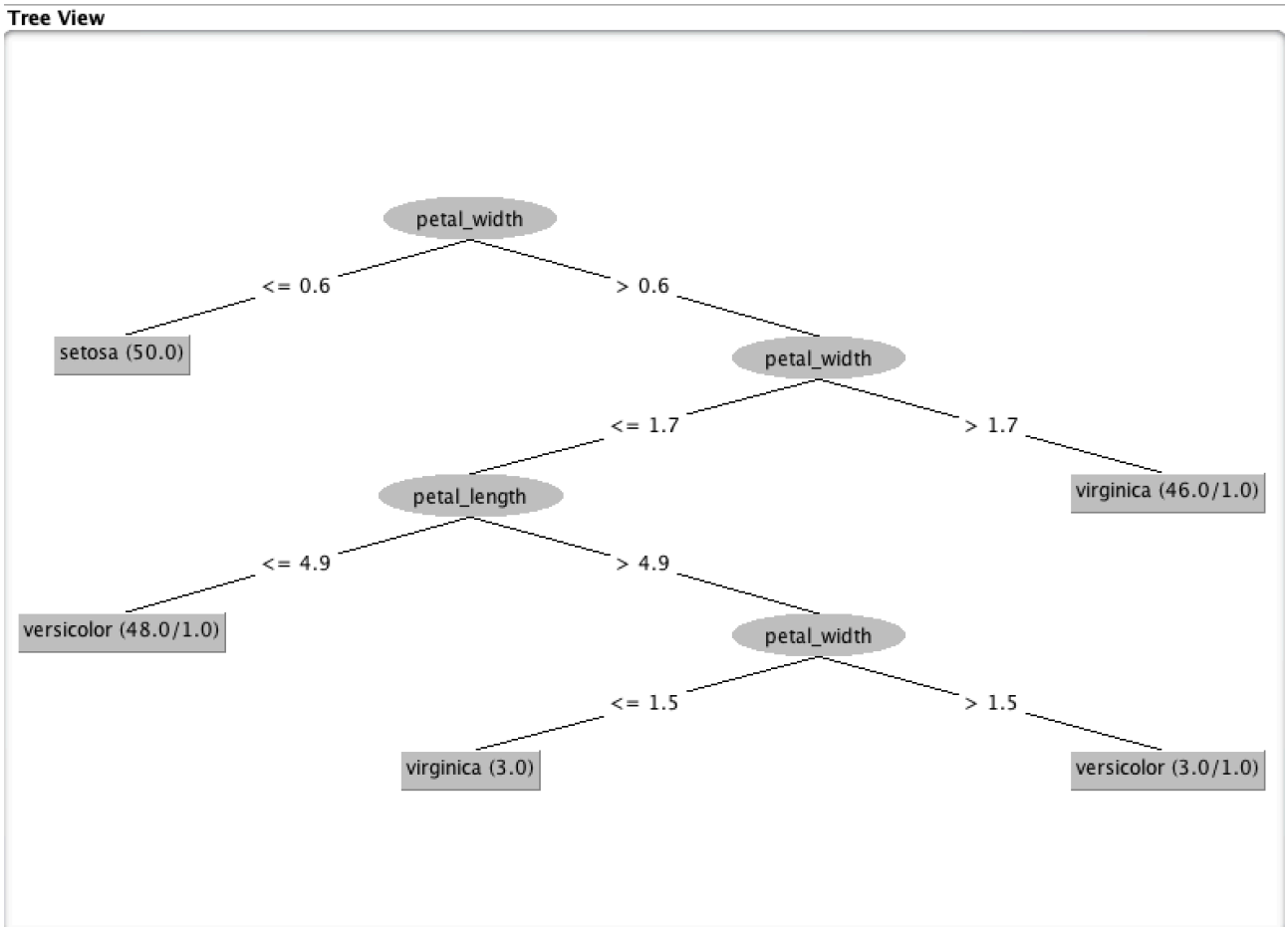


Figura 4 - árvore de decisão formada pela divisão dos dados ao meio.

O segundo experimento com a divisão de 80% treino e 20% teste obteve um resultado melhor do que o experimento anterior, atingindo uma acurácia de 100%. A matrix de confusão é mostrada na figura 5 e a árvore derivada do experimento é mostrada na figura 6.

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 11  0  0 |  a = setosa
  0 10  0 |  b = versicolor
  0  0  9 |  c = virginica
```

Figura5 - Matrix de confusão do experimento 80% treino e 20% teste

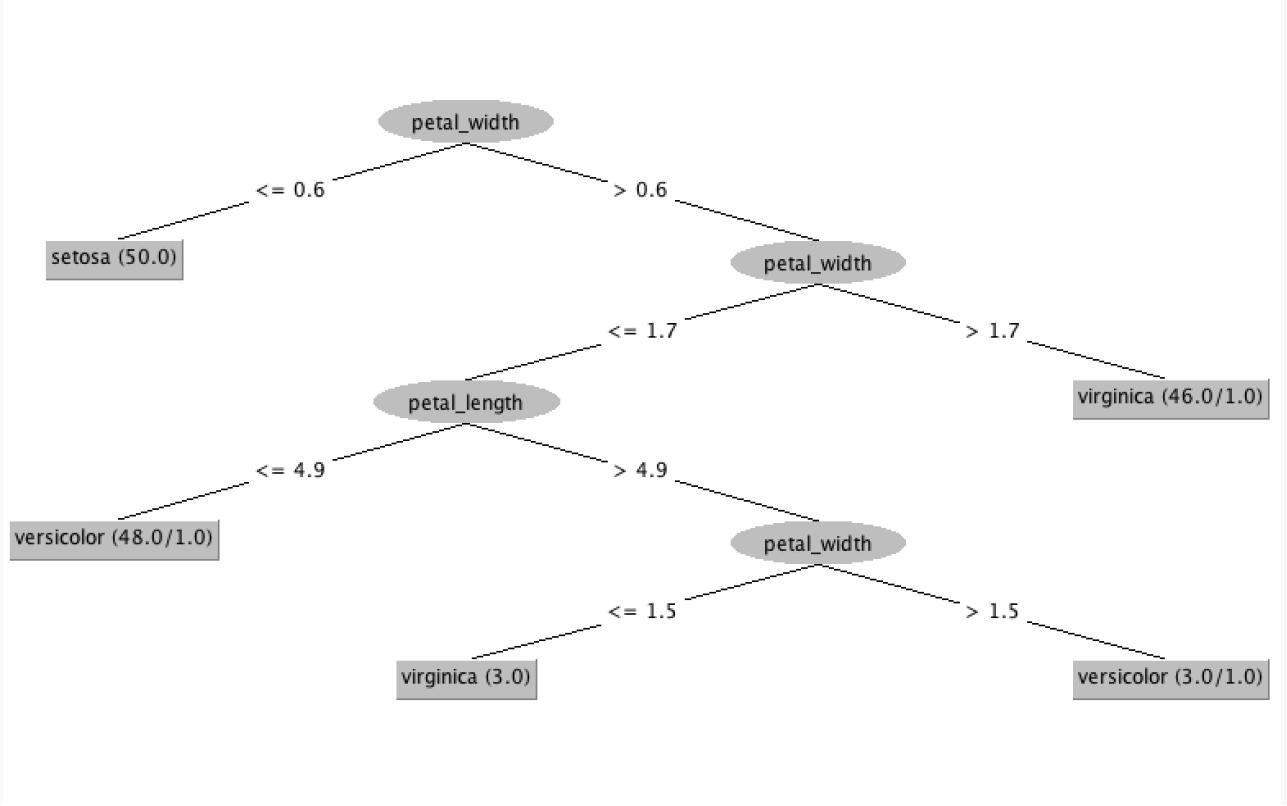
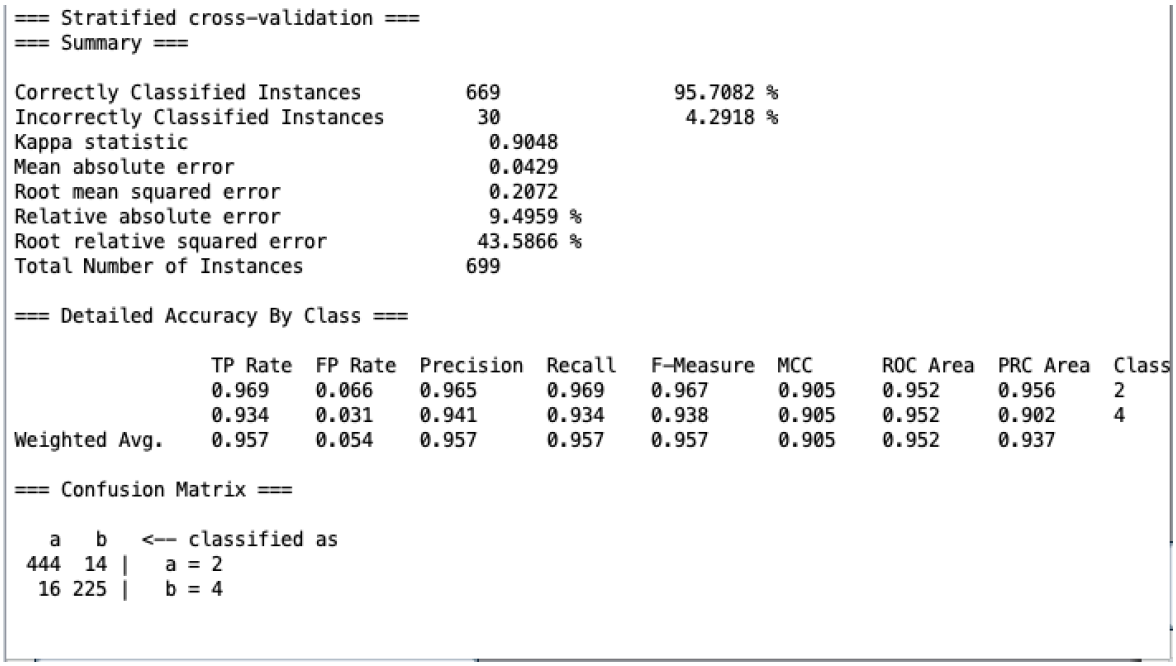


Figura 6 - árvore de decisão derivada do experimento com 80% dos dados de treino e 20% para teste

Questão 3

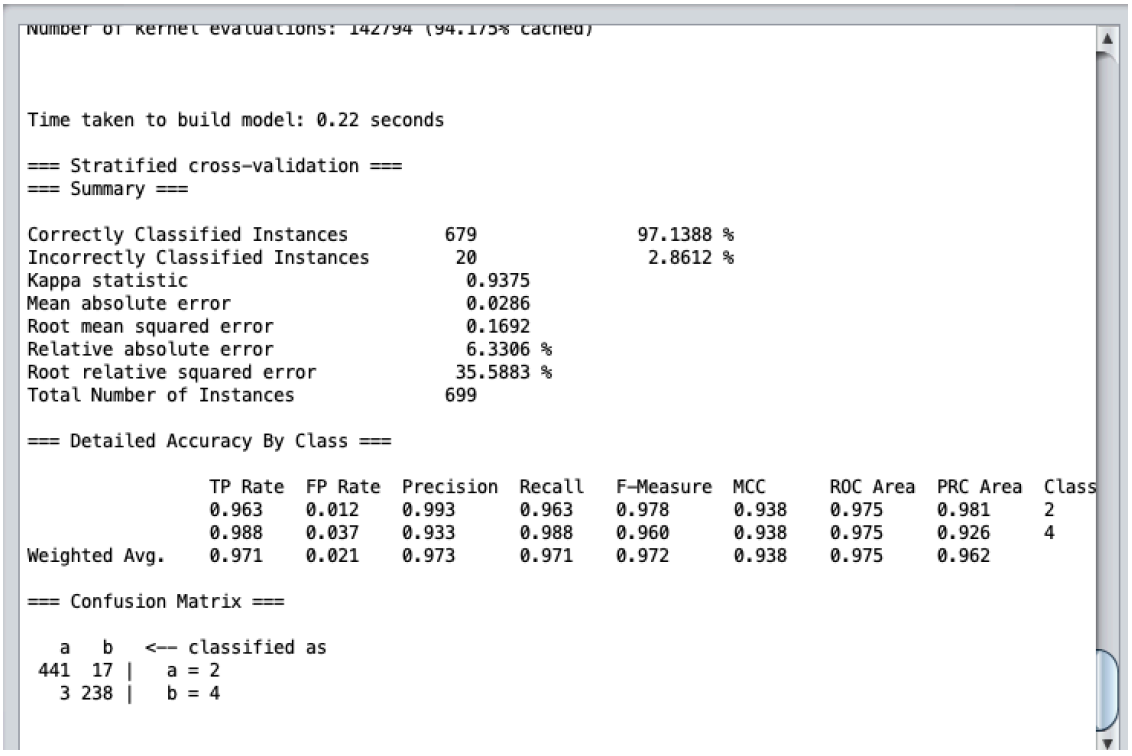
Questão 4

Os dados escolhidos para a realização desta questão foi o conjunto de dados referentes a cancer de mama de Wisconsin. Ele está disponível no UC Irvine Machine Learning Repository recomendado na primeira questão.
O *dataset* escolhido possui 699 amostras, cada uma com 11 características a serem analisadas, com a rotulagem da lesão ser maligna ou benigna. Das amostras encontradas nos dados 458 são de lesões benignas e 241 malignas, e para caracterizar uma lesão em alguma das duas categorias 10 características são analisadas para que possa chegar a uma predição de diagnóstico.
Para a realização do exercício foi mantido o a configuração de 10-fold cross validation para duas configurações de *kernels: o* polinomial e o de Base Radial. Para cada configuração foi testado o parâmetro C em três diferentes valores: 1.0, 0.5 e 0.25. Os resultados das respectivas configurações é mostrado nas figuras a seguir.



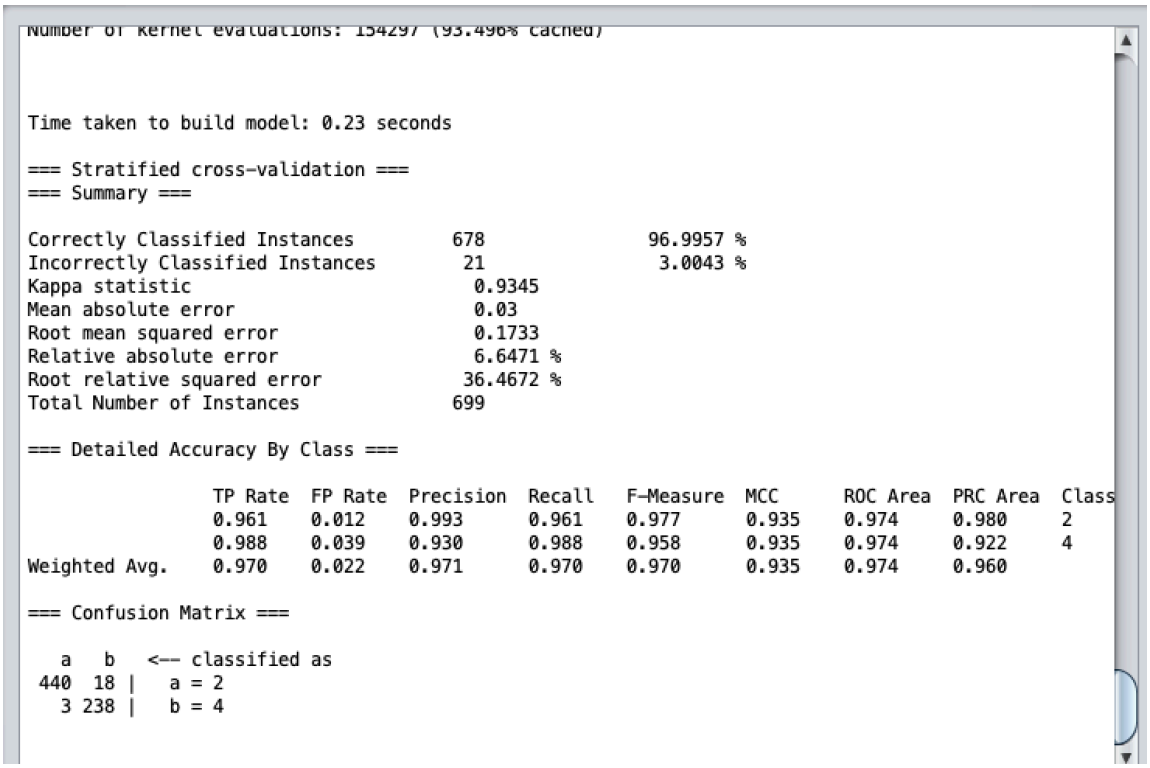
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         669               95.7082 %
Incorrectly Classified Instances        30                4.2918 %
Kappa statistic                          0.9048
Mean absolute error                      0.0429
Root mean squared error                  0.2072
Relative absolute error                  9.4959 %
Root relative squared error             43.5866 %
Total Number of Instances              699

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.969    0.066    0.965      0.969    0.967      0.905    0.952     0.956     2
                 0.934    0.031    0.941      0.934    0.938      0.905    0.952     0.902     4
Weighted Avg.    0.957    0.054    0.957      0.957    0.957      0.905    0.952     0.937

=== Confusion Matrix ===

   a   b   <-- classified as
 444  14 |   a = 2
  16 225 |   b = 4
```

Figura 7 - configuração de a função de kernel polinomial com o parâmetro C = 1.0



```
Number of kernel evaluations: 142794 (94.175% cached)



Time taken to build model: 0.22 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         679               97.1388 %
Incorrectly Classified Instances        20                2.8612 %
Kappa statistic                          0.9375
Mean absolute error                      0.0286
Root mean squared error                  0.1692
Relative absolute error                  6.3306 %
Root relative squared error             35.5883 %
Total Number of Instances              699

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.963    0.012    0.993      0.963    0.978      0.938    0.975     0.981     2
                 0.988    0.037    0.933      0.988    0.960      0.938    0.975     0.926     4
Weighted Avg.    0.971    0.021    0.973      0.971    0.972      0.938    0.975     0.962

=== Confusion Matrix ===

   a   b   <-- classified as
 441  17 |   a = 2
   3 238 |   b = 4
```

Figura 8 - configuração de função de kernel polinomial com o parâmetro C = 0.5

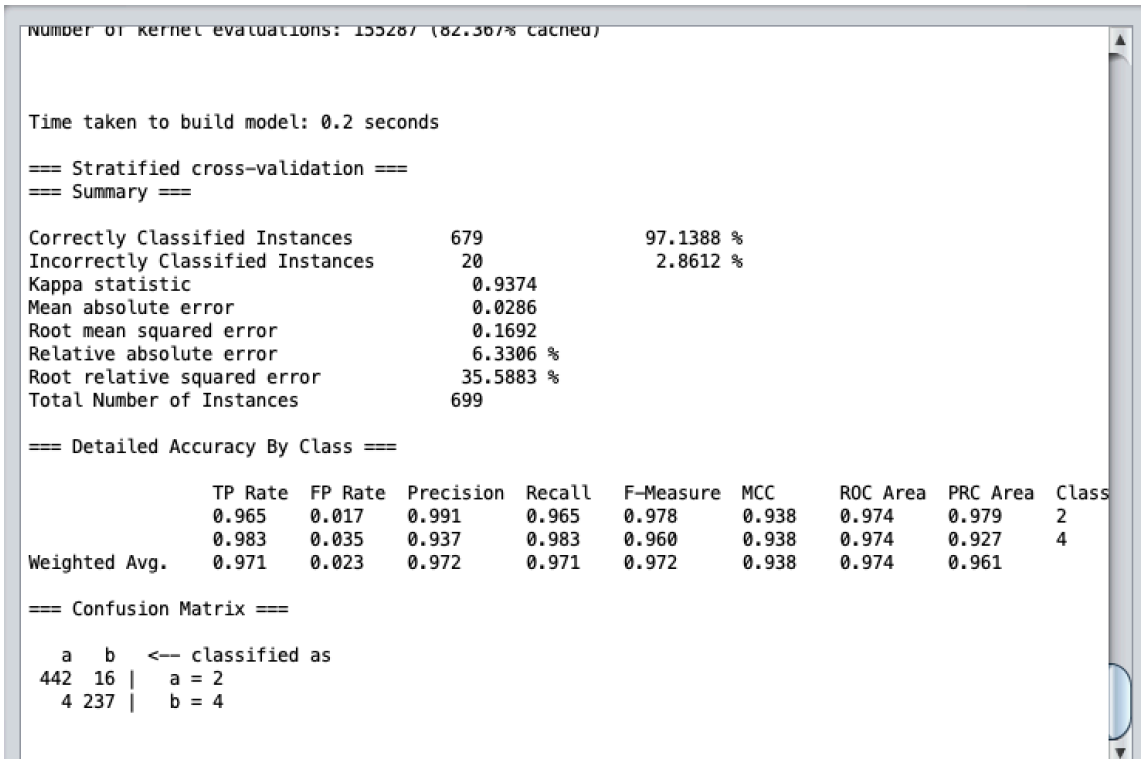


```
Number of kernel evaluations: 154297 (93.496% cached)



Time taken to build model: 0.23 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         678               96.9957 %
Incorrectly Classified Instances        21                3.0043 %
Kappa statistic                          0.9345
Mean absolute error                      0.03
Root mean squared error                  0.1733
Relative absolute error                  6.6471 %
Root relative squared error             36.4672 %
Total Number of Instances              699

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.961    0.012    0.993      0.961    0.977      0.935    0.974     0.980     2
                 0.988    0.039    0.930      0.988    0.958      0.935    0.974     0.922     4
Weighted Avg.    0.970    0.022    0.971      0.970    0.970      0.935    0.974     0.960

=== Confusion Matrix ===

   a   b   <-- classified as
 440  18 |   a = 2
   3 238 |   b = 4
```

Figura 9 - configuração de função de kernel polinomial com o parâmetro C = 0.25

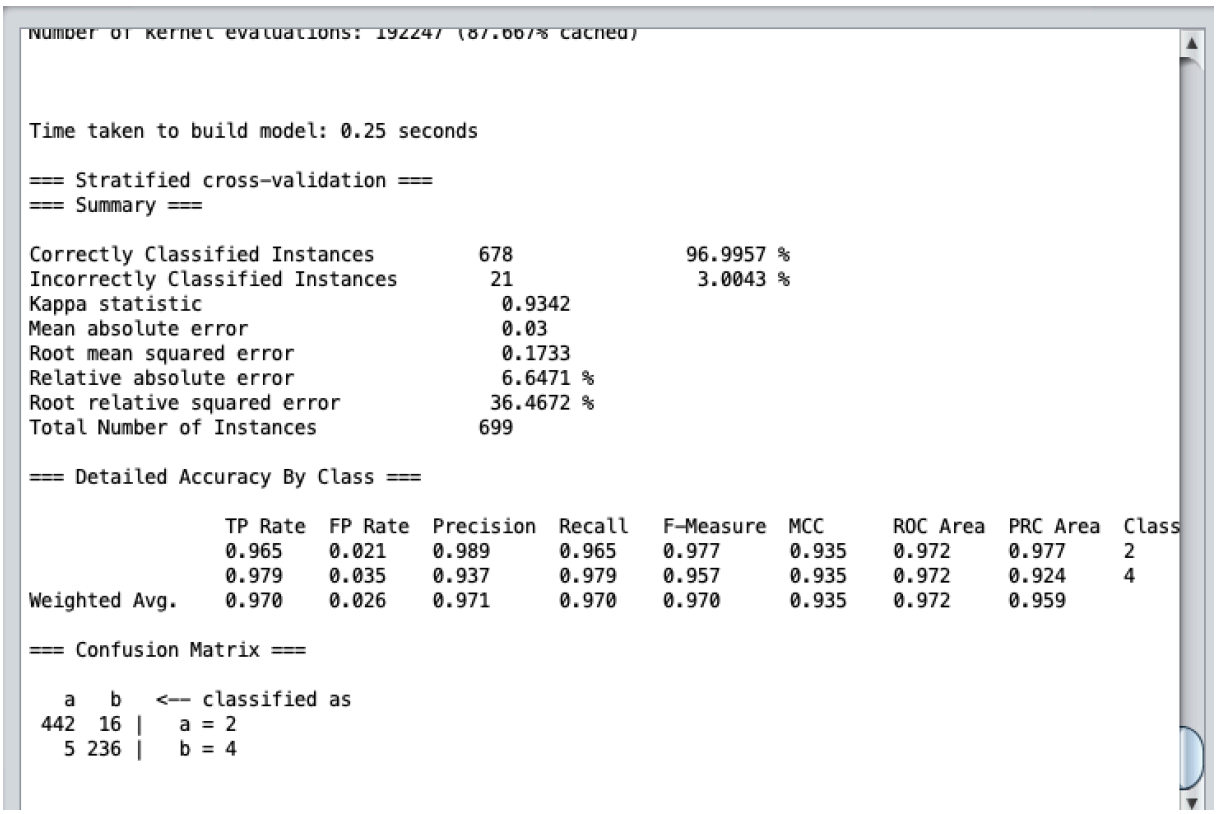
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         680               97.2818 %
Incorrectly Classified Instances        19                2.7182 %
Kappa statistic                          0.9406
Mean absolute error                      0.0272
Root mean squared error                  0.1649
Relative absolute error                  6.0141 %
Root relative squared error             34.6872 %
Total Number of Instances              699

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.965    0.012    0.993      0.965    0.979      0.941    0.976     0.981     2
                 0.988    0.035    0.937      0.988    0.962      0.941    0.976     0.930     4
Weighted Avg.    0.973    0.020    0.974      0.973    0.973      0.941    0.976     0.964

=== Confusion Matrix ===

   a   b   <-- classified as
 442  16 |   a = 2
   3 238 |   b = 4
```

Figura 10 - configuração de função de Kernel com base radial com o parâmetro C = 1.0



```
Number of kernel evaluations: 155287 (82.367% cached)



Time taken to build model: 0.2 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         679               97.1388 %
Incorrectly Classified Instances        20                2.8612 %
Kappa statistic                          0.9374
Mean absolute error                      0.0286
Root mean squared error                  0.1692
Relative absolute error                  6.3306 %
Root relative squared error             35.5883 %
Total Number of Instances              699

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.965    0.017    0.991      0.965    0.978      0.938    0.974     0.979     2
                 0.983    0.035    0.937      0.983    0.960      0.938    0.974     0.927     4
Weighted Avg.    0.971    0.023    0.972      0.971    0.972      0.938    0.974     0.961

=== Confusion Matrix ===

   a   b   <-- classified as
 442  16 |   a = 2
   4 237 |   b = 4
```

Figura 11 - configuração de função de Kernel com base radial com o parâmetro C = 0.5



```
Number of kernel evaluations: 192247 (87.667% cached)



Time taken to build model: 0.25 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         678               96.9957 %
Incorrectly Classified Instances        21                3.0043 %
Kappa statistic                          0.9342
Mean absolute error                      0.03
Root mean squared error                  0.1733
Relative absolute error                  6.6471 %
Root relative squared error             36.4672 %
Total Number of Instances              699

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.965    0.021    0.989      0.965    0.977      0.935    0.972     0.977     2
                 0.979    0.035    0.937      0.979    0.957      0.935    0.972     0.924     4
Weighted Avg.    0.970    0.026    0.971      0.970    0.970      0.935    0.972     0.959

=== Confusion Matrix ===

   a   b   <-- classified as
 442  16 |   a = 2
   5 236 |   b = 4
```

Figura 12 - configuração de função de Kernel com base radial com o parâmetro C = 0.25

É possível verificar que dentre as configurações obtidas a que possuiu o melhor resultado em relação as métricas foi a de kernel polinomial com o parâmetro C de 1.0 (Figura 10). A separação das classes foi melhor obtida com essa configuração.

Acesse o Link do projeto no. GitHub para mais detalhes