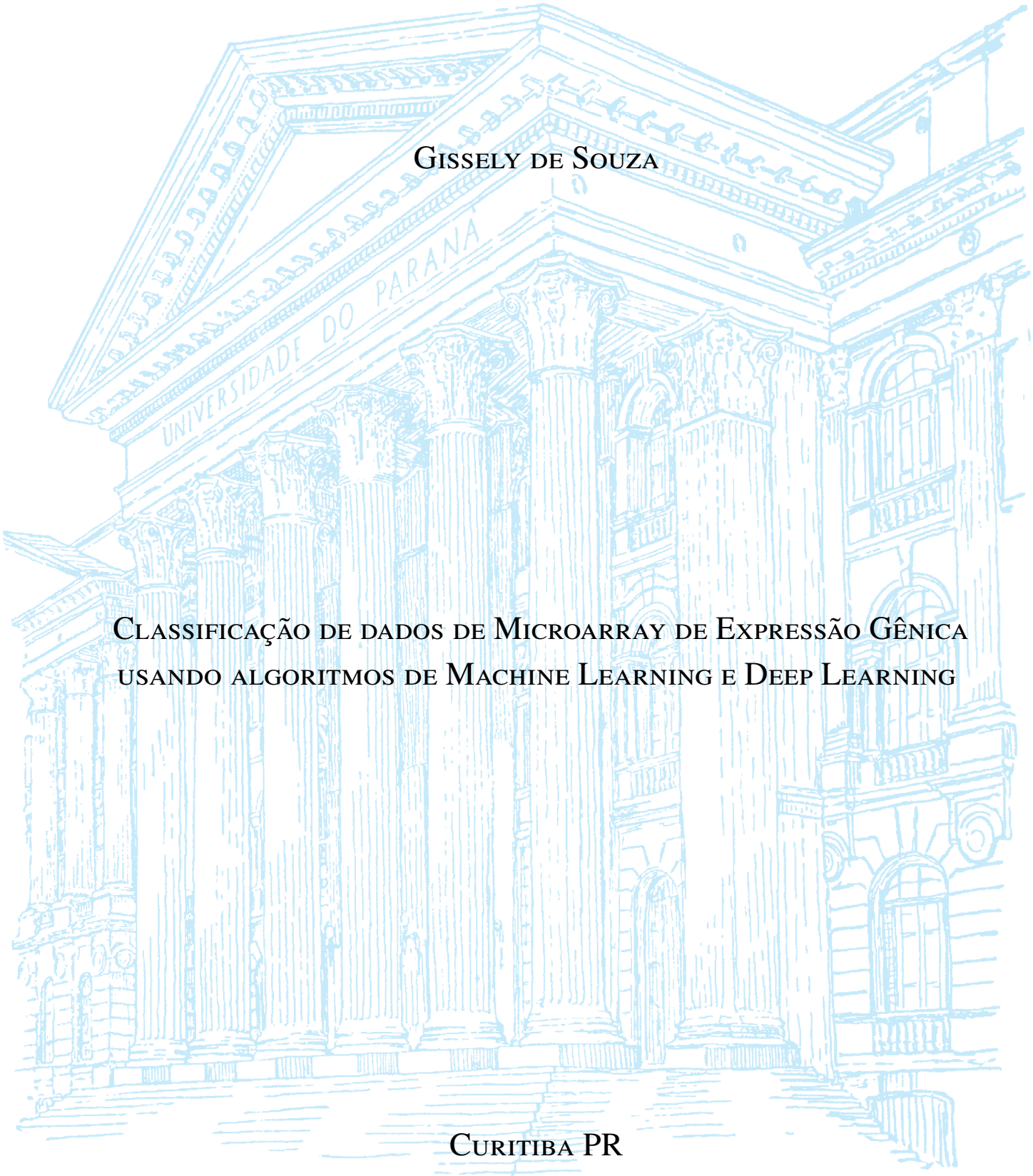
Universidade Federal do Paraná

Gissely de Souza



Classificação de dados de Microarray de Expressão Gênica usando algoritmos de Machine Learning e Deep Learning

Curitiba PR
2021

Gissely de Souza

Classificação de dados de Microarray de Expressão Gênica usando algoritmos de Machine Learning e Deep Learning

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Informática Biomédica, Setor de Ciências Exatas, da Universidade Federal do Paraná..

Área de concentração: *Informática Biomédica*.

Orientador: Eduardo J. Spinosa.

Curitiba PR
2021

A alguém...

Agradecimentos

Ao meu país, o Brasil, que tanto amo, e que garante a minha oportunidade de acesso à Educação Pública. À Universidade Federal do Paraná, que foi minha segunda casa. Aos meus professores, mestres da minha formação. Ao meu orientador Eduardo J. Spinosa. Aos meus pais que já não estão mais aqui, a minha família que me deu todo o suporte incluindo o mais novo membro o meu esposo David Westbrook.

Resumo

Os métodos de Aprendizagem de Máquina e Aprendizagem Profunda para a classificação do câncer são um importante tópico de pesquisa na medicina, pois como uma ferramenta de auxílio ao diagnóstico médico permite soluções e resultados bem-sucedidos em tratamentos médicos.

Para analisar diferentes tumores, ou do mesmo tipo, mas com características diferentes, foram usados conjuntos de dados de *microarray* rotulados com câncer que normalmente, denominada a propriedade, contém um pequeno número de amostras e um grande número de níveis de expressão de genes denominados como a alta dimensionalidade, e ainda, as bases usadas apresentam-se como multi-classes e outras com classes desbalanceadas que tornam a classificação bastante desafiadora. Neste estudo, são comparados os algoritmos clássicos mais amplamente usados em Aprendizagem de Máquina e Aprendizagem Profunda para classificar os diferentes tipos tumores descritos nas bases de dados de *microarray*.

Palavras-chave: Análise de Expressão Gênica, Classificação, Aprendizado de Máquina, Aprendizado profundo.

Abstract

Machine Learning and Deep Learning methods for cancer classification are an important research topic in medicine because as an aiding tool for medical diagnosis it allows successful solutions and results in medical treatments.

To analyze different tumors, or of the same type, but with different characteristics, *microarray* datasets labeled with cancer were used, which normally, called the property, contain a small number of samples and a large number of expression levels of genes called high dimensionality, and yet, the bases used are presented as multi-classes and others with unbalanced classes that make the classification quite challenging. In this study, the most widely used classical algorithms in Machine Learning and Deep Learning to classify the different types of tumors described in *microarray* databases are compared.

Keywords: Gene Expression Analysis, Classification, Deep Learning, Machine Learning.

Lista de Figuras

3.1 (a) Representação de *underfitting*, (b) *overfitting* e (c) *fitting*. 23

3.2 À esquerda resolução algoritmo supervisionado e à direita algoritmo não supervisionado. Fonte:Feita pelo autor 24

3.3 Etapas a serem realizadas em algoritmos KNN. Fonte: Retirado de https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn 26

3.4 (À esquerda) Duas classes são mostradas usando duas cores diferentes. SVM encontra a melhor linha que separa as duas classes. (À direita) Os pontos mais próximos do limite de decisão são chamados de vetores de suporte. O SVM encontra o limite de decisão maximizando sua distância dos vetores de suporte. Fonte: Retirado de https://www.linkedin.com/pulse/find-panda-exploring-support-vector-machines-lee-schlenker 27

3.5 Função do kernel do SVM. Fonte: Retirado de https://www.linkedin.com/pulse/find-panda-exploring-support-vector-machines-lee-schlenker 27

3.6 O algoritmo de floresta aleatória usa várias árvores de decisão (geradas aleatoriamente) para gerar a saída final. Fonte: Retirado de https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from-scratch-with-real-24ff198a1c57 28

3.7 Curva de função logística. Fonte: Retirado de https://matheusfacure.github.io/2017/02/25/regr-log/ 29

3.8 Um neurônio biológico em comparação com uma rede neural artificial: (a) neurônio humano; (b) neurônio artificial; (c) sinapse biológica; e (d) sinapses de RNA. Fonte: (Suzuki, 2013) 30

3.9 Representação da estrutura de um único perceptron, junto com sua posição dentro de um perceptron multicamadas em grande escala. Fonte: Adaptado pelo autor https://github.com/PetarV-/TikZ/blob/master 32

3.10 Convolução, ativação e subamostragem são realizadas iterativamente nos mapas de características para criar gradualmente um mapa de atributo global. O mapa de atributo final obtido pela repetição desse processo várias vezes é inserido em uma camada totalmente conectada, ou seja, uma rede neural *feedforward* para realizar tarefas de classificação. Fonte: Retirado de https://medium.datadriveninvestor.com/convolutional-neural-network-cnn-simplified-ecafd4ee52c5 e adaptada pelo autor. 34

3.11 Distribuição balanceadas das classes nas partições da CV *Kfold* estratificada. Fonte: (Zeng e Martinez, 2000) 35

3.12 Exemplo de CV aninhada (aqui 5x2 CV aninhada), ou seja no *loop* externo a divisão é em 5 *Kfold* e *loop* interno 2 *Kfold*. Fonte: Desenho inspirado em https://www.programmersought.com/article/66124607365/ 36
3.13 Etapas envolvidas no experimento de *microarray*. 39
3.14 Três níveis de processamento de dados de expressão gênica de*microarray*. Fonte: (Bolón-Canedo et al., 2014). 40
3.15 Etapas modulares da expressão gênica incluindo a transcrição, *splicing* de RNA, tradução e modificação pós-tradução de um determinado produto gênico sintetizado (proteína). Fonte: (Morange, 2009) 41

5.1 Matriz de tamanho $m \times n$ e as classes são um vetor de tamanho $m \times 1$, onde m é o número de amostras e n é o número de genes para cada classe. As características compõem a representação por algum valor de x e as classes pelos valores de y. Fonte: Elaborada pelo autor e inspirada em (Mandal e Mukhopadhyay, 2013) . . 46
5.2 Divisão dos dados em quatro partições, cada uma com o método *Holdout* Fonte: Elaborado pelo autor. 48
5.3 Arquiteturas da Rede Neural Artificial usadas para classificação de câncer. Fonte: Elaborado pelo autor e inspirada em (Tabares-Soto et al., 2020) e (Shanmugamani, 2018). 50

6.1 Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA 54
6.2 À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes 54
6.3 Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA 55
6.4 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes 55
6.5 Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA 56
6.6 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes 56
6.7 Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA 57
6.8 À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes 57
6.9 Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA 58
6.10 Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA 58
6.11 À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta. 59

6.12 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. 59
6.13 Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA 59
6.14 À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta. 60
6.15 À esquerda, o comportamento da *accuracy* do treinamento e sua validação, à direita a função de perda para o treinamento e validação. 60
6.16 Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA 61
6.17 Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA 63
6.18 À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes 63
6.19 Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA 64
6.20 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes 64
6.21 Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA 65
6.22 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes 65
6.23 Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA 66
6.24 À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes 66
6.25 Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA 67
6.26 Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA 68
6.27 À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta. 68
6.28 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. 69
6.29 Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA 69
6.30 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. 70
6.31 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. 70
6.32 Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA 71

6.33 Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA . . . 72
6.34 À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes . . . 73
6.35 Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA . . . 73
6.36 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes . . . 74
6.37 Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA . . . 74
6.38 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes . . . 75
6.39 Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA . . . 75
6.40 À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes . . . 76
6.41 Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA . . . 76
6.42 Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA . . . 77
6.43 À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta. . . . 77
6.44 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. . . . 78
6.45 Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA . . . 78
6.46 À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta. . . . 79
6.47 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. . . . 79
6.48 Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA . . . 80
6.49 Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA . . . 81
6.50 À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes . . . 82
6.51 Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA . . . 82
6.52 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes . . . 83
6.53 Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA . . . 83

6.54 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes 83
6.55 Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA 84
6.56 À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes 84
6.57 Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA 85
6.58 Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA 86
6.59 À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta. 86
6.60 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. 87
6.61 Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA 87
6.62 À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta. 88
6.63 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. 88
6.64 Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA 89
6.65 Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA 90
6.66 À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes 91
6.67 Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA 91
6.68 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes 92
6.69 Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA 92
6.70 À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes 93
6.71 Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA 93
6.72 À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes 94
6.73 Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA 94
6.74 Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA 95

6.75 À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta. 95
6.76 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. 96
6.77 Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA 96
6.78 À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta. 97
6.79 À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação. 97
6.80 Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA 98

Lista de Tabelas

5.1 Resumo dos conjuntos de dados e suas características, como tipo de chip microarray (Chip), (Tecido), número de classes (Classes), número de amostras (m), distribuição de amostras dentro das classes (Dist. Classes) e dimensionalidade após pré-processamento (d) e dimensionalidade original da base (n). 47

5.2 Parâmetros dos algoritmo de ML testados, onde m é o número de amostras. . . . 49

5.3 Parâmetros dos algoritmos das Redes Neurais 51

6.1 Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão 54

6.2 Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão 55

6.3 Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão 56

6.4 Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão 57

6.5 Scores dos algoritmos e seus respectivos parâmetros na base de dados Câncer de Pulmão 58

6.6 Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão 58

6.7 Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão 59

6.8 Parâmetros dos algoritmos das Redes Neurais para a base de dados Câncer de Pulmão 61

6.9 Scores dos algoritmos com o método validação cruzada estratificada na base de dados Câncer de Pulmão 61

6.10 Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados Câncer de Pulmão 62

6.11 Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata 63

6.12 Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata 64

6.13 Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata 65

6.14 Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata 66

6.15 Scores dos algoritmos e seus respectivos parâmetros na base de dados Câncer de Próstata 67
6.16 Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata 68
6.17 Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata 69
6.18 Parâmetros dos algoritmos das Redes Neurais para a base de dados Câncer de Próstata 71
6.19 Scores dos algoritmos com o método validação cruzada estratificada na base de dados Câncer de Próstata 71
6.20 Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados Câncer de Próstata 72
6.21 Predição com as métricas da matriz de confusão com a base de dados 11_Tumores 72
6.22 Predição com as métricas da matriz de confusão com a base de dados 11_Tumores 73
6.23 Predição com as métricas da matriz de confusão com a base de dados 11_Tumores 74
6.24 Predição com as métricas da matriz de confusão com a base de dados 11_Tumores 75
6.25 Scores dos algoritmos e seus respectivos parâmetros na base de dados 11_Tumores 76
6.26 Predição com as métricas da matriz de confusão com a base de dados 11_Tumores 77
6.27 Predição com as métricas da matriz de confusão com a base de dados 11_Tumores 78
6.28 Parâmetros dos algoritmos das Redes Neurais para a base de dados 11_Tumores . 80
6.29 Scores dos algoritmos com o método validação cruzada estratificada na base de dados 11_Tumores 80
6.30 Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados 11_Tumores. 81
6.31 Predição com as métricas da matriz de confusão com a base de dados Leucemia . 81
6.32 Predição com as métricas da matriz de confusão com a base de dados Leucemia . 82
6.33 Predição com as métricas da matriz de confusão com a base de dados Leucemia . 83
6.34 Predição com as métricas da matriz de confusão com a base de dados Leucemia . 84
6.35 Scores dos algoritmos e seus respectivos parâmetros na base de dados Leucemia. 85
6.36 Predição com as métricas da matriz de confusão com a base de dados Leucemia . 86
6.37 Predição com as métricas da matriz de confusão com a base de dados Leucemia . 87
6.38 Parâmetros dos algoritmos das Redes Neurais para a base de dados Leucemia . . 89
6.39 Scores dos algoritmos com o método validação cruzada estratificada na base de dados Leucemia 89
6.40 Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados Leucemia 90
6.41 Predição com as métricas da matriz de confusão com a base de dados SRBCT . . 90
6.42 Predição com as métricas da matriz de confusão com a base de dados SRBCT . . 91
6.43 Predição com as métricas da matriz de confusão com a base de dados SRBCT . . 92

6.44 Predição com as métricas da matriz de confusão com a base de dados SRBCT . . 93
6.45 Scores dos algoritmos e seus respectivos parâmetros na base de dados SRBCT. . 94
6.46 Predição com as métricas da matriz de confusão com a base de dados SRBCT . . 95
6.47 Predição com as métricas da matriz de confusão com a base de dados SRBCT . . 96
6.48 Parâmetros dos algoritmos das Redes Neurais para a base de dados SRBCT . . . 98
6.49 Scores dos algoritmos com o método validação cruzada estratificada na base de dados SRBCT 98
6.50 Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados SRBCT 99

Lista de Acrônimos

ANN	Artificial Neural Networks
CNN	Convolutional Neural Network
CV	Cross-Validation
DL	Deep Learning
DNA	Deoxyribonucleic Acid
DNN	Deep Neuronal Networks
IA	Artificial Intelligence
KNN	K-Nearest Neighbors
ML	Machine Learning
MLP	Multi-Layer Perceptron
PCA	Principal Component Analysis
RF	Random Forest
RL	Logistic Regression
SVM	Support Vector Machine

Sumário

1 Introdução 18
1.1 Motivação 19
1.2 Estrutura do trabalho 20
2 Proposta 21
2.1 Objetivos 21
2.1.1 *Objetivos Específicos* 21
3 Fundamentação Teórica 22
3.1 Processo de Aprendizagem 22
3.2 Algoritmos de Aprendizagem de Máquina 23
3.2.1 *Aprendizagem Supervisionada* 23
3.2.2 *Aprendizagem Não Supervisionada* 24
3.3 Classificação 24
3.4 K-Nearest Neighbors (KNN) 25
3.5 Support Vector Machines (SVM) 26
3.6 Random Forest (RF) 28
3.7 Regressão logística (RL) 29
3.8 Redes Neurais e Algoritmos de Aprendizado Profundo 29
3.9 Multilayer Perceptron (MLP) 30
3.9.1 Estrutura do aprendizado da MLP 32
3.10 Redes Neurais Convolucionais (CNN) 32
3.10.1 Estrutura do aprendizado da CNN 33
3.11 Pré-processamento de Dados 34
3.11.1 *Normalização* 34
3.11.2 *Análise de Componentes Principais (PCA)* 34
3.12 Validação Cruzada (CV) 35
3.12.1 *Validação Cruzada Estratificada* 35
3.12.2 *Nested Cross-validation* 35
3.13 Métricas de Avaliação 36
3.14 Dados de Microarray 38
3.15 Análise de Expressão Gênica 40
4 Revisão Bibliográfica 42
4.1 Processo de Aprendizagem em Pesquisa de Bioinformática 42

4.2 Predição de Dados de Expressão Gênica no Estado da Arte 42
5 Materiais e métodos 45
5.1 *Bases de Dados* 45
5.2 *Descrição dos conjuntos de dados* 46
5.3 Métodos 47
5.4 Preparação dos dados 47
5.5 Processo de Execução e Ajuste dos algoritmos 48
5.6 Implementação da Rede Neural 49
5.7 Validação cruzada 51
6 Resultados e Discussão 53
6.1 Algoritmos com a base de dados Câncer de Pulmão 54
6.2 Algoritmos com a base de dados Câncer de Próstata 62
6.3 Algoritmos com a base de dados 11_Tumores 72
6.4 Algoritmos com a base de dados Leucemia 81
6.5 Algoritmos com a base de dados SRBCT 90
6.6 Resumo dos Resultados 100
7 Conclusões 102
Referências 104

1 Introdução

Câncer é um termo genérico para um grande grupo de doenças que podem afetar qualquer parte do corpo e é uma das doenças mais letais para a saúde humana, causada pela proliferação rápida anormal de células, levando a malformações malignas ou tumores de características patológicas diversas (Pavlidis e Fizazi, 2009). Fatores como falha na divisão celular e danos ao Ácido Desoxirribonucleico DNA são os principais fatores internos, enquanto a exposição a substâncias como produtos químicos na fumaça do tabaco, radiação e raios ultravioleta do sol são fatores ambientais significativos (Su et al., 2001). Dependendo de como o gene se expressa em estruturas de proteínas, adicionando estes fatores, podem levar ao desenvolvimento de células de forma carcinogênicas (Su et al., 2001). Uns dos caminhos mais promissores para a diminuição da mortalidade por câncer é a detecção precoce e o tratamento da doença já instalada no organismo, o que exige uma melhor compreensão dos mecanismos moleculares subjacentes ao surgimento e progressão do tumor (Loomans-Kropp e Umar, 2019).

Os dados de expressão gênica são dados genômicos de sequência de DNA representativos e amplamente usados para prever o prognóstico de pacientes com câncer (Sotiriou et al., 2006). Os dados do nível de expressão do gene são dados numéricos que indicam o grau de ativação de um gene que codifica a informação da proteína.

No caso do diagnóstico de doenças com base em uma grande quantidade de dados genômicos ou proteômicos a tecnologia de *microarray* se destaca como um método de análise que pode processar grandes quantidades de informações em um curto espaço de tempo no sentido de monitorar simultaneamente o nível de expressão de milhares de genes. Além disto, este método pode identificar mutações de expressão gênica. Ao realizar as análises genéticas moleculares em dados de *microarray* são identificadas alterações, ou assinaturas, que discriminam características biológicas que pode discernir as respostas a diversos tratamentos. Isso pode permitir a previsão de doenças genéticas e um tratamento preciso, portanto, uma maior eficácia e redução dos efeitos colaterais do tratamento Krief et al. (1999).

No campo da Biologia Computacional, Genômica, estatística e na classificação de padrões, a análise de dados de expressão gênica é uma das pesquisas de domínios desafiadoras.

Podemos pesquisar e desenvolver modelos que preveem com mais confiabilidade estes dados de tal maneira que o computador aprenda com os dados. Estes modelos são alimentados por informações que estão na forma de dados numéricos como por exemplo, valor de expressão do gene ou seja, alimentado por dados biológicos advindos de bases de *microarray*. Assim que o computador vai sendo alimentado, ele vai aprendendo com estes dados e cria o modelo do seu aprendizado. O modelo será capaz de prever se tal amostra é de um específico tipo de câncer, ou se a amostra não contém câncer. Este processo é denominado Aprendizagem de Máquina ML supervisionada onde cada amostra é rotulada e enviada para um algoritmo que classificará estes dados de câncer no nível de expressão gênica. Com tudo, o principal desafio com a análise de dados de câncer de *microarray* é associado à natureza da dimensionalidade de dados (o número de instâncias é consideravelmente menor que a dimensionalidade do problema, ou seja, o número de características (genes) é muito maior que o número de instâncias (amostras)). Além disso,

existem outros impedimentos experimentais que tornam a análise de dados de *microarray* um domínio fascinante. Os conjuntos de dados médicos são tipicamente ruidosos, têm variações nos valores das características, alguns se apresentam como multiclasses e outros podem conter um número desequilibrado de classes, o que pode resultar em *overfitting* e menor precisão de classificação (Wang et al., 2014).

A classe desbalanceada ocorre quando temos um conjunto de dados que possui muitos exemplos de uma classe e poucos exemplos da outra classe. Na situação desse modelo, a classificação pode ser enviesada, ou seja, ele tende a classificar os novos dados como sendo da classe que possui mais exemplos. Assim como, o número de pessoas diagnosticadas com uma determinada doença é bem menor que o número de pessoas sem a doença (Chandra e Gupta, 2011). Com estas questões levantadas, o problema requer a construção cuidadosa de um modelo de previsão sobre os dados fornecidos. E também, uma boa estratégia de pré-processamento nos dados e uma técnica de aprendizagem pode minimizar ou solucionar estes problemas.

Estudos anteriores foram realizados para a identificação e classificação da expressão gênica em dados de *microarray*. Muito destas pesquisas utilizam os mais tradicionais algoritmos de ML e Redes Neurais Artificial ANNs. Alguns destes estudos alcançaram resultados animadores para analisar conjuntos de dados em grande escala (Bengio et al., 2013). Com esses avanços, o aprendizado profundo DL alcançou um desempenho de ponta em uma ampla gama de aplicações, incluindo a bioinformática e genômica (Yue e Wang, 2018), análise de amostras metagenômicas (Ceballos et al., 2019) , identificação de elementos somáticos transponíveis em câncer de ovário ((Tang et al., 2017), identificação e classificação de retrotransposons em plantas (Orozco-Arias et al., 2019) e classificação de câncer usando Análise de Componentes Principais PCA (Kavitha et al., 2018). Muitas abordagens foram usadas, muitos autores alcançaram alta precisão nessas publicações como a comparação dos algoritmos de ML e DL em (Tabares-Soto et al., 2020). No entanto, há falta de uma comparação mais detalhada incluindo várias métricas de avaliação para os métodos de classificação entre os algoritmo de ML e DL. Neste estudo utilizaremos alguns dos tradicionais algoritmos de ML como Máquina de Vetor de Suporte SVM, K Vizinho mais Próximo KNN, Regressão Logística LR, Floresta Aleatória RF e também, algoritmos de DL como as Redes Neurais Artificiais Perceptron Multicamadas MLP e Rede Neural Convolucional CNN, todos usando a técnica supervisionada para classificação. Para validar o método proposto, consideramos 5 conjuntos de dados de câncer de *microarray* padrão, como: próstata, pulmão, leucemia e dois conjunto de dados múltiplos tecidos. O pré-processamento dos valores dos atributos são normalizados usando o processo de dimensionamento das amostras com a abordagem *Mini_Max_Scaler* e a redução de dimensionalidade de dados utiliza a análise de componente principal PCA. E para validar o modelo , Nested Cross-validation estratificada para os algoritmos de ML e validação cruzada estratificada para comparar todos os algoritmos deste experimento.

1.1 Motivação

Ao analisar o conjunto de dados de *microarray*, os pesquisadores são confrontados com sérios desafios das técnicas computacionais, devido à natureza da dimensionalidade de dados (poucas amostras com muitos atributos). Muitas técnicas clássicas de classificação geralmente apresentam dificuldades para lidar com esses dados pesados. Além disso, existem outros aspectos na análise de dados de *microarray* como por exemplo, amostras multiclasses e dados como amostras desbalanceadas. A classificação supervisionada de amostras de tecido em tais problemas de alta dimensão é difícil, mas muitas vezes crucial para o diagnóstico e tratamento bem-sucedido. Alguns pesquisadores propuseram fazer o pré-processamento de dados de genes antes da classificação do câncer. Realizar a seleção de genes ajuda a reduzir o tamanho dos dados,

melhorando assim o tempo de execução. Mais importante, a seleção de genes pode remover um grande número de genes irrelevantes, o que melhora a precisão da classificação. Além da seleção do gene, existem várias questões relacionadas à classificação do câncer que são de grande preocupação aos pesquisadores. Essas questões são derivadas do contexto biológico do problema, e a importância médica do resultado. Estudos têm mostrado que mudanças na expressão gênica estão relacionadas com diferentes tipos de câncer. O que motiva estudos para experimentos multi-classes embora, algoritmos de classificação típicos podem ser tendenciosos para as classes com o maior número de amostras, uma vez que as regras de classificação para prever essas instâncias altas são favorecidas positivamente na precisão do desempenho, enquanto ignoram as regras de previsão que preveem amostras de classes minoritárias e, às vezes considerá-los como ruído, uma vez que algoritmos de classificação preferem regras mais gerais. No entanto, para amenizar este problema, uma boa aplicação de seleção de atributos e métodos de validação cruzada estratificada tornaria mais fácil de lidar com estes obstáculos. Embora várias abordagens de classificação diferentes tenham sido utilizadas para lidar com tais problemas de classificação, isto desperta motivação de explorar os método de ML e principalmente, DL para comparar a performance neste tipo de situação desafiadora.

1.2 Estrutura do trabalho

O Capítulo 2 apresentará a proposta, objetivos e objetivos específicos. No Capítulo 3 deste trabalho serão apresentados os fundamentos e definições necessárias para o entendimento do trabalho. O Capítulo 4 elucidará os trabalhos relacionados e a revisão de literatura do problema, e abordará alguns trabalhos que envolvem análise de expressão gênica com ML e DL. Após isso, o capítulo 5 descreve a metodologia dos experimentos realizados neste trabalho, que envolvem a implementação dos algoritmos de ML, DL e as técnicas utilizadas para o melhoramento do desempenho dos algoritmos para uma eficiente classificação. Visto isso, apresentamos os resultados experimentais presentes no Capítulo 6 que serão demonstrados e discutidos e por fim, serão apresentadas as conclusões deste trabalho no Capítulo 6.

2 Proposta

2.1 Objetivos

Este trabalho se propõe a fazer experimentos com diferentes bases de dados de *microarray* com tipos de câncer de diferentes tecidos. Essas bases ou conjuntos de dados se diferem pelo tipo de a tecnologia do chip, pelo tipo de tecidos, número de amostras, número de classes e bases desbalanceadas por classe. Além disto estas bases têm a característica de ser de alta dimensionalidade o qual necessita de alguns ajustes e refinamento nos dados para poder extrair características com mais fidelidade em busca de uma correta classificação dos tipos de câncer.

2.1.1 *Objetivos Específicos*

Dentre os objetivos específicos, deseja-se analisar o impacto dos classificadores de ML e DL e suas comparação. Os seguintes cenários serão considerados:

- Deseja-se analisar o efeito dos algoritmos ML e DL e suas comparações em diferentes situações.
- Busca também métodos de pré-processamento que obtenham um melhor desempenho com as bases desbalanceadas e de alta dimensionalidade.
- Aplicação dos algoritmos em diferentes bases de dados, com natureza, número de instâncias, número de classes e dimensionalidades diferentes.
- Refinar os algoritmos com diferentes parâmetros que se encaixem para um bom desempenho.

3 Fundamentação Teórica

Nesta seção, serão apresentados os conceitos e fundamentos necessários para a interpretação do funcionamento dos algoritmos de ML e DL. Inicialmente, na Seção 3, é apresentado o processo de aprendizagem para ML e DL, classificação e os conceitos básicos dos algoritmos de ML e DL. Na seção 3.11 são apresentadas as técnicas e definições de pré-processamento do dados e na seção 3.12, validação cruzada. Na seção 2.13 são demostradas as métricas para cálculo dos resultados. E por fim na seção 2.14 é descrita a obtenção de dados de *microarray* e na seção 2.15 a sua análise da expressão do gene.

3.1 Processo de Aprendizagem

Para entender sobre ML e DL precisamos nos situar onde estes termos se encontram. Eles são um ramo da Inteligência Artificial (*Artificial Intelligence*), a abreviação em inglês de AI e foi desenvolvida a partir da ciência da computação moderna, iniciada em meados de 1950 pelo matemático britânico Alan Turing (2004) e pelo matemático húngaro-americano Von Neumann (1988). Desde então, surgiram muitas maneiras de estudar inteligência artificial. No entanto, todos os métodos têm uma coisa em comum; tratam a inteligência de uma perspectiva computacional, ou seja, têm entrada e saída de informações como computadores. E no campo da ciência e tecnologia, muitas descobertas e invenções estão aprendendo com a natureza, assim como a inteligência artificial que é geralmente definida como a simulação da inteligência humana por máquinas. Um exemplo disso, os neurônios são as unidades básicas dos cérebros biológicos e podem ser considerados a fonte da inteligência biológica. Ao imitar neurônios, a rede neural artificial é um modelo matemático que simula a rede neural do cérebro humano em termos de estrutura, mecanismo e função. Em síntese, simula o cérebro humano. E cada neurônio é uma unidade de processamento de informações com múltiplas entradas e saídas únicas. O principal objetivo de qualquer tarefa de ML é otimizar o desempenho do modelo não apenas nos dados de treinamento, mas também em conjuntos de dados adicionais de teste. Quando um modelo aprendido exibe esse comportamento, considera-se que generaliza bem. Com este objetivo, as bases de dados são divididas aleatoriamente em pelo menos dois subconjuntos:

- *Treinamento:* A amostra de dados usada para ajustar o modelo.
- *Teste:* A amostra de dados usada para fornecer uma avaliação imparcial de um ajuste do modelo final no conjunto de dados de treinamento.

Em seguida, um modelo tão complexo quanto possível é aprendido pela máquina (conjunto de treinamento), a parte não treinada (conjunto de teste) é verificada com o modelo para ver o desempenho de generalização. Este processo é crucial para evitar *overfitting* ou *underfitting* (Zou et al., 2018).

- *Overfitting* se refere ao 'sobreajuste' que acontece quando um modelo aprende os detalhes e o ruído nos dados de treinamento ao ponto de impactar negativamente o desempenho do modelo em novos dados. Isso significa que o ruído ou flutuações aleatórias nos dados de treinamento são captados e aprendidos como conceitos pelo modelo. O problema é que esses conceitos não se aplicam a novos dados e impactam negativamente a capacidade de generalização dos modelos.

- *Underfitting* refere-se a um modelo que não pode modelar os dados de treinamento nem generalizar para novos dados. É um modelo insuficiente, inadequado e terá um desempenho ruim nos dados de treinamento.

Portanto, um algoritmo de aprendizado sólido deve atingir um equilíbrio apropriado entre a flexibilidade do modelo e a quantidade de dados de treinamento.

Esta ideia é vista na Figura 3.1, onde o melhor modelo para o algoritmo é quando ele acha um meio termo que é a generalização dos dados representada pela letra (c) na Figura3.1.

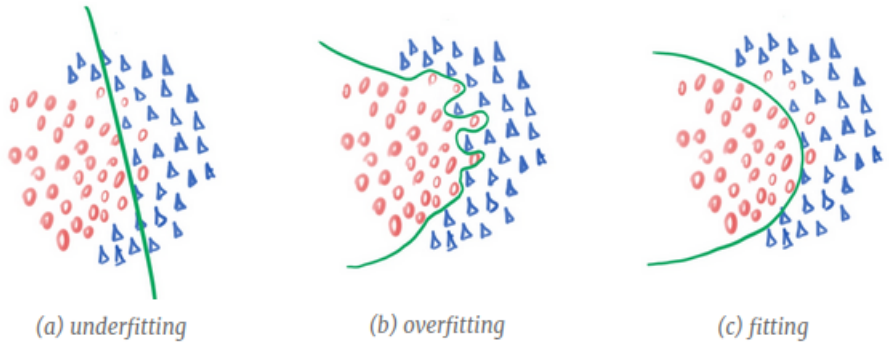


Figura 3.1: (a) Representação de *underfitting*, (b) *overfitting* e (c) *fitting*

3.2 Algoritmos de Aprendizagem de Máquina

O algoritmo de classificação é um componente importante da tecnologia de mineração de dados e aprendizado de máquina ML, parte de seu objetivo principal é classificar exemplos desconhecidos de acordo com o modelo de classificação de amostra de treinamento.

ML é dividido principalmente em três abordagens: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço.

Neste trabalho foi utilizado somente o aprendizado supervisionado, mas relataremos a aprendizagem não supervisionada para uma melhor compreensão sobre a classificação que é o foco deste trabalho.

3.2.1 *Aprendizagem Supervisionada*

Algoritmos de aprendizado supervisionado são projetados para aprender por exemplo. É usado quando um participante humano sabe a resposta para um problema e deseja treinar uma IA para descobri-la. É como ensinar um algoritmo para a 'resposta correta' com a ajuda de um professor. O treinamento de um algoritmo de aprendizado supervisionado requer a construção de um conjunto de entrada a ser emparelhado com uma saída específica. O algoritmo então procura padrões dentro do conjunto de entrada e os correlaciona com a saída. Com base nesses dados de treinamento, o algoritmo de aprendizado supervisionado pode pegar entradas invisíveis e decidir

quais rótulos atribuir. O objetivo dos algoritmos de aprendizagem supervisionada é classificar e compreender corretamente os novos dados de entrada, prevendo rótulos apropriados.

3.2.2 *Aprendizagem Não Supervisionada*

Ao contrário da aprendizagem supervisionada, o conjunto de dados na aprendizagem não supervisionada é completamente sem rótulo. Sendo necessário classificar o conjunto amostral de acordo com a similaridade entre as amostras na tentativa de minimizar as distâncias entre as classes e maximizar as distâncias entre os *clusters*. Em aplicações práticas, em muitos casos, é impossível saber o rótulo da amostra com antecedência, ou seja, não há classes correspondente à amostra de treinamento, então a configuração do algoritmo só pode ser aprendido a partir dos dados definido sem o rótulo da amostra. Em geral, o algoritmo busca encontrar padrões nos próprios dados. Embora o aprendizado não supervisionado tenha a vantagem de não precisar fornecer um rótulo, tem a desvantagem de ser difícil avaliar o desempenho do modelo porque não há rótulo.

Os algoritmos comuns incluem: Autocodificador Esparso, Análise de Componente Principal PCA, algoritmo *K-MEANS* (K-médias), *DBSCAN* (baseado em densidade) entre outros.

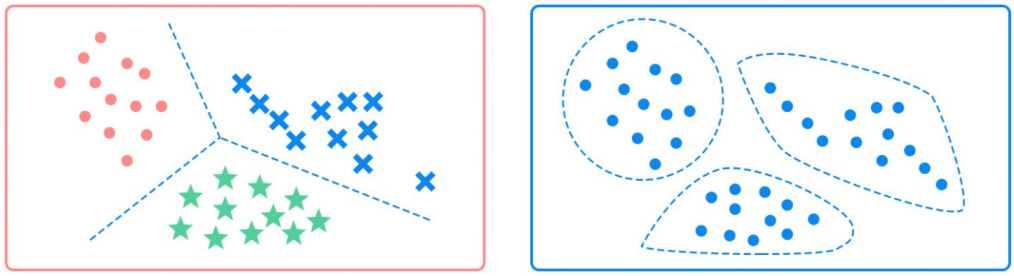


Figura 3.2: À esquerda resolução algoritmo supervisionado e à direita algoritmo não supervisionado. Fonte:Feita pelo autor

Os métodos de aprendizagem supervisionada devem ter conjuntos de treinamento e amostras de teste. Encontra o padrão no conjunto de treinamento e usa-o na amostra de teste. Por outro lado, a aprendizagem não supervisionada não possui um conjunto de treinamento, apenas um conjunto de dados. Um exemplo de como estes dois métodos se comportam é visto na Figura 3.2.

3.3 Classificação

A classificação é uma subcategoria da aprendizagem supervisionada. O objetivo é prever os rótulos de classe de uma nova amostra com base em observações anteriores. Um classificador buscará regras que lhe permitirá atribuir a amostras desconhecidas para qualquer uma das classes como por exemplo: diagnosticar a célula doente (Quackenbush, 2001). Dentro a tarefa de classificação, é necessário um número suficiente de amostras para permitir que o algoritmo seja capaz de usar estas amostras e treine com eficiência e utilize outras amostras independentes que não foram usadas para testar o algoritmo que servirá como conjunto de teste. O algoritmo funciona por meio de um modelo ou função que aprende a mapear valores dos atributos para as classes rotuladas. Este é um tipo de tarefa preditiva. Existem muitos algoritmos de classificação disponíveis na literatura, como Artificial Neural Network, Árvores de Decisão,

KNN, SVM e muitos outros. O mais popular, que é a linha de base padrão em problemas de classificação, é o KNN, onde as decisões de classificação são feitas por meio de uma votação e o treinamento do mecanismo e do modelo armazena o conjunto de dados de forma que as consultas possam ser feitas eficientemente. Outra família de métodos de classificação compreende os chamados modelos lineares, para os quais um algoritmo de aprendizagem estima tantos pesos quanto recursos dos dados de treinamento, de modo que a previsão de classificação é feita como uma função do produto escalar entre os pesos e uma amostra de teste. Os modelos lineares são rápidos para treinar, prever e também escalam bem para conjuntos de dados nos quais o número de atributos é grande em comparação com o número de amostras. Os métodos lineares que foram testados são o SVC, RF e LR. Ao contrário dos modelos lineares, árvores de decisões e RF são invariáveis para o dimensionamento de dados e funcionam bem com atributos em escalas diferentes, sendo assim, uma boa razão para testá-los.

Os Algoritmos de DL também podem ser usados para classificação. MLPs são adequadas para dados não lineares, enquanto CNNs automatizam a tarefa cara de extração de características; uma tarefa inevitável nas abordagens clássicas de ML.

3.4 K-Nearest Neighbors (KNN)

A ideia central de KNN (K-vizinho mais próximo) é que se a maioria das K amostras mais próximas no espaço de características de uma amostra pertencem a uma determinada classe, então a amostra também pertence a esta classe, e tem as características da amostra nesta categoria. Para este método o algoritmo faz a determinação da decisão de classificação por algum método de cálculo de menor distância entre as instâncias das classes que são: (Distância Euclidiana, Distância de Hamming, Distância Manhattan, Distância de Markowski) (Prasatha et al., 2017). O algoritmo KNN pode ser usado não apenas para classificação, mas também para regressão. Apesar de ser um algoritmo simples, possui a desvantagem de ser lento e pode se tornar fraco à medida que o número de atributos aumenta. Também é difícil determinar o valor ideal de K que é o número de vizinhos usados.

As etapas de um algoritmo KNN são:

- Recebe um dado não classificado;
- Mede a distância do novo dado com todos os outros dados que já estão classificados;
- Obtém o parâmetro K vizinhos mais próximo;
- Verifica a classe de cada instância dos K vizinhos mais próximos e conta a quantidade de cada classe que aparece;
- Toma como resultado a classe que mais apareceu dentre os dados que tiveram as menores distâncias;
- Atribui ao novo dado a classe tomada como resultado da classificação.

A Figura 3.12 mostra os passos para classificar uma nova instância, que neste caso para K=3 vizinhos mais próximos é classificada para a Classe B.

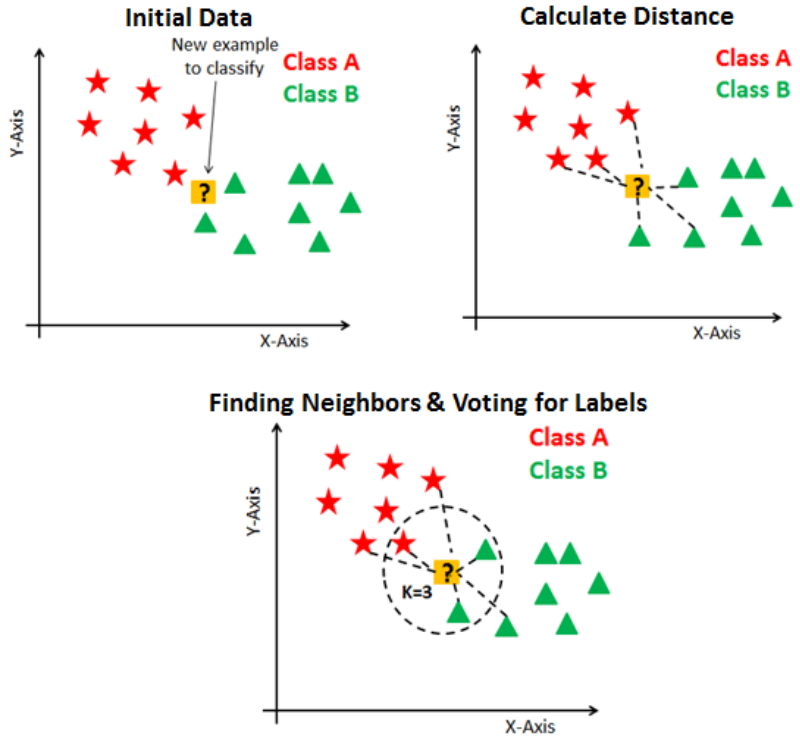


Figura 3.3: Etapas a serem realizadas em algoritmos KNN. Fonte: Retirado de https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

3.5 Support Vector Machines (SVM)

Máquina de vetores de suporte SVM é um tipo de regressor e classificador linear generalizado de acordo com a aprendizagem supervisionada. O modelo padrão é formulado para problemas de classificação binária, e problemas multiclasse são normalmente reduzidos a uma série de problemas binários. O objetivo do algoritmo SVM é encontrar a melhor linha ou limite de decisão (este melhor limite de decisão é chamado de hiperplano) que pode separar o espaço N-dimensional (N o número de atributos) em classes para que possamos colocar facilmente o novo ponto de dados na categoria correta no futuro. O SVM escolhe os pontos extremos que ajudam na criação do hiperplano. Ele busca maximizar a margem ou a distância entre os pontos mais próximos da fronteira de decisão gerada. Os pontos mais próximos da fronteira de decisão são denominados vetores de suporte. Os vetores de suporte são pontos de dados que estão mais próximos do hiperplano e influenciam a posição e a orientação do hiperplano. Margem significa a largura máxima da linha paralela ao hiperplano que não possui pontos de dados internos. Apenas para problemas linearmente separáveis o algoritmo pode encontrar tal hiperplano.

Em problemas de classificação, muitas vezes existem várias soluções. Conforme mostrado no lado esquerdo da Figura 3.4, no caso de separabilidade linear ideal, haverá vários planos de decisão.

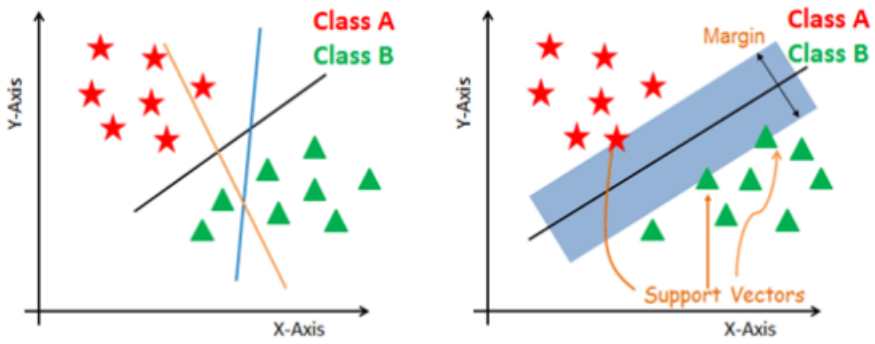


Figura 3.4: (À esquerda) Duas classes são mostradas usando duas cores diferentes. SVM encontra a melhor linha que separa as duas classes.
(À direita) Os pontos mais próximos do limite de decisão são chamados de vetores de suporte. O SVM encontra o limite de decisão maximizando sua distância dos vetores de suporte.
Fonte: Retirado de https://www.linkedin.com/pulse/find-panda-exploring-support-vector-machines-lee-schlenker

As máquinas de vetores de suporte são poderosas em situações em que o número de atributo da classe (colunas) é maior do que o número de amostras (linhas). Também é eficaz em dimensões altas (como dados de *microarray*) e eficiente em termos de memória porque usa um subconjunto de um conjunto de dados para aprender os vetores de suporte. As imagens mostradas aqui têm uma classe que é linearmente separável, mas às vezes as classes não podem ser separadas por uma linha reta na dimensão atual. Um SVM é capaz de mapear os dados em uma dimensão superior de forma que se tornem separáveis por uma margem. Ao resolver o problema linear inseparável, ele pode resolver habilmente a operação do produto interno no espaço de alta dimensão, introduzindo a função *kernel*, de modo a resolver bem o problema de classificação não linear. Conforme mostrado na Figura 3.5, por meio da introdução da função *kernel*, os dados linearmente inseparáveis são mapeados para um espaço de características de dimensão superior, onde o conjunto de treinamento é separável como mostrado abaixo:

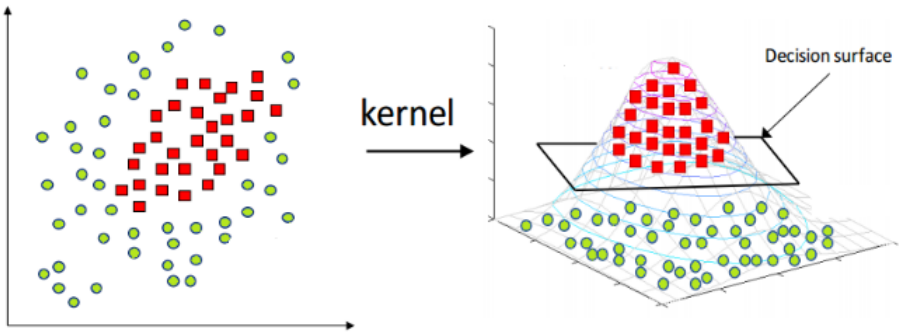


Figura 3.5: Função do kernel do SVM.
Fonte: Retirado de https://www.linkedin.com/pulse/find-panda-exploring-support-vector-machines-lee-schlenker

3.6 Random Forest (RF)

A floresta aleatória - RF é um algoritmo de aprendizado de máquina baseado em árvore que usa o poder de várias árvores de decisão para tomar decisões. Como o nome sugere, é uma “floresta” feita de árvores, e é usada principalmente em regressão e classificação. E cada árvore de decisão pode ser simplesmente entendida como uma série de decisões para alcançar um determinado resultado. Uma árvore de busca é construída de forma que os nós folhas representam prováveis classes do problema, e os demais nós representam valores relacionados às características do problema que são utilizados para a comparação. A partir da raiz, os atributos da instância que se deseja classificar são comparados com os valores dos nós não folhas (relacionados a uma característica), e de acordo com o resultado da comparação o fluxo da busca é alterado para algum dos ramos do nó. De modo geral, uma árvore simples não produz resultados eficazes. Com isto, a opção do algoritmo de RF se torna interessante. É uma floresta aleatória por razão de que árvores de decisão são criadas aleatoriamente. Cada nó na árvore de decisão é um subconjunto aleatório de atributos da classe, que é usado para calcular a saída. E integra a saída de uma única árvore de decisão para gerar o resultado final da saída. Há muitas árvores de decisão na floresta, e não há correlação entre cada árvore de decisão. Depois de obter a floresta, quando uma nova amostra de entrada chega, cada árvore de decisão na floresta faz um julgamento separadamente para ver a qual classe a amostra pertence (para o algoritmo de classificação). A floresta aleatória reduzirá a parte da variância do erro em vez da parte do viés, portanto, em um determinado conjunto de dados de treinamento, a árvore de decisão pode ser mais precisa do que uma floresta aleatória, mas em um conjunto de dados de validação inesperado, a floresta aleatória em geral vence em termos de precisão.

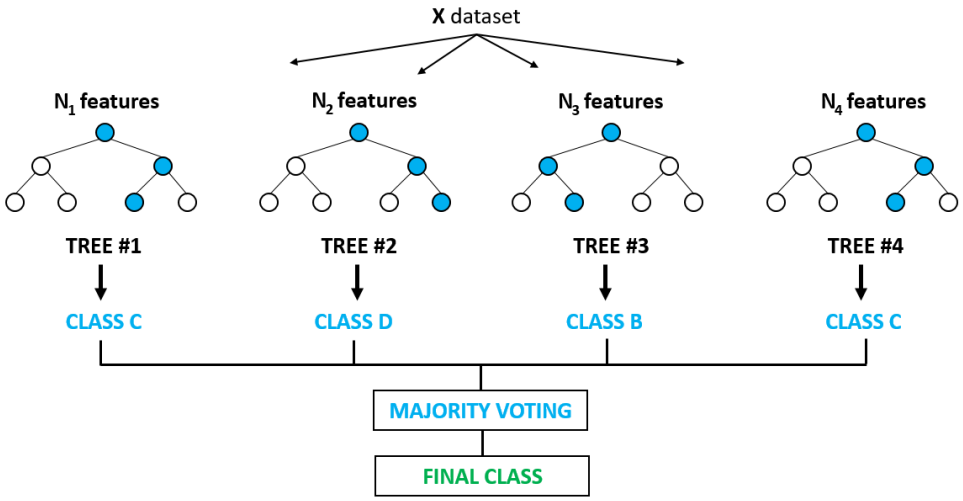


Figura 3.6: O algoritmo de floresta aleatória usa várias árvores de decisão (geradas aleatoriamente) para gerar a saída final.
Fonte: Retirado de https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from-scratch-with-real-24ff198a1c57

3.7 Regressão logística (RL)

A regressão logística é um algoritmo de aprendizado supervisionado que usa a regressão para prever a probabilidade dos dados pertencentes a uma determinada classe como um valor entre 0 e 1 e classificá-los como relacionado a esta classe com valor de probabilidade.

Por exemplo, um classificador de *spam*. Quando um e-mail é recebido e se a probabilidade de ser *spam* for 0,5 ou mais, ele é classificado como *spam*, e se a probabilidade for inferior a 0,5, é classificado como *não spam*. Ou seja, ao prever se um evento ocorrerá, a ocorrência do evento é classificada como 1 e a saída aparece na forma de probabilidade da classe padrão. Para isto, a RL transforma os resultados usando a função de ativação *Sigmóide* e a probabilidade se expressa como uma curva entre 0 e 1 como pode ser visto na Figura 3.7.

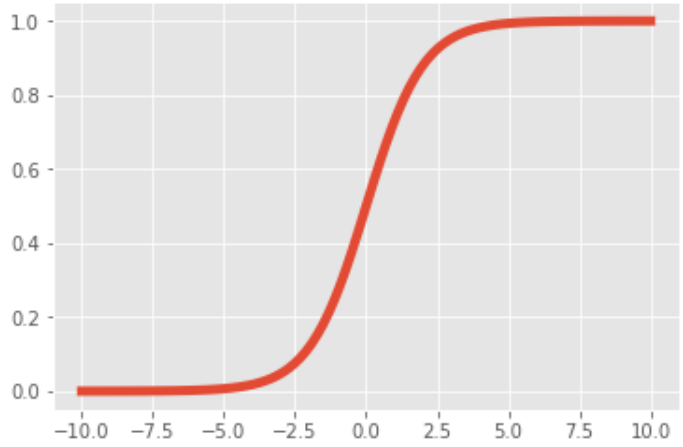


Figura 3.7: Curva de função logística.
Fonte: Retirado de https://matheusfacure.github.io/2017/02/25/regr-log/

Para gerar a saída (valor y) é convertida logaritmicamente para o (valor x) , usando a função logarítmica.

$$y = \frac{1}{1 + e^{-x}}$$

Esta equação converte qualquer número real em uma probabilidade entre 0 e 1 com base no conjunto de limiar. Em seguida, aplica-se um limite para forçar essa probabilidade na classificação binária.

3.8 Redes Neurais e Algoritmos de Aprendizado Profundo

O aprendizado profundo usa redes neurais artificiais que permitem que as máquinas tomem decisões precisas sem ajuda humana. O DL é considerado uma evolução e é um método importante entre muitos métodos de ML , estas que são o subcampo da IA que por sua vez tem muitas razões para seu desenvolvimento:

- A evolução dos algoritmos de aprendizagem profunda;
- Aumento explosivo no volume de dados;
- Avanços nas GPUs.

Os algoritmos de ML começaram com a rede neural *feedforward* mais simples. Ele imita como os neurônios em nosso cérebro estão conectados camada por camada. Uma rede neural

feedforward também é chamada de rede neural artificial.

• *Redes Neurais Biológicas*

Geralmente, refere-se a uma rede composta de neurônios cerebrais biológicos, células, contatos, etc. No sistema nervoso humano, células especiais chamadas neurônios carregam impulsos nervosos, axônios transmitem neurônios do corpo celular para outras células e dendritos conectam várias células em forma de ramos. Os neurônios atuam como entradas para a sinapse. As sinapses se conectam entre as células. As sinapses são conectadas de acordo com estímulos externos, como neurônios visuais ou neurônios da pele, e essas conexões permitem o aprendizado (Eluyode e Akomolafe, 2013).

• *Redes neurais artificiais*

A Rede Neural Artificial (Artificial Neural Network) ANN é um sistema de processamento de informações projetado para imitar a estrutura e a função do cérebro humano. É uma rede composta pela interconexão de neurônios artificiais. Ela abstrai e simplifica as funções do cérebro humano. É uma parte importante da simulação da inteligência humana. Essa abordagem reflete várias características básicas das funções cerebrais humanas, como processamento paralelo de informações, aprendizagem, associação, classificação de padrões e memória.

A relação entre as redes neurais biológica e artificial é demostrada na Figura 3.8.

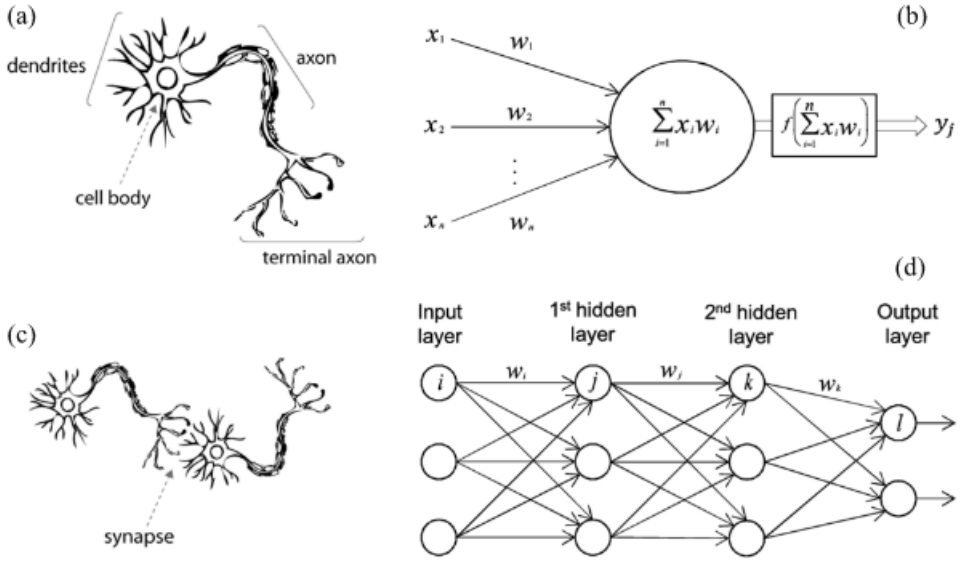


Figura 3.8: Um neurônio biológico em comparação com uma rede neural artificial: (a) neurônio humano; (b) neurônio artificial; (c) sinapse biológica; e (d) sinapses de RNA. Fonte: (Suzuki, 2013)

3.9 Multilayer Perceptron (MLP)

MLP é uma técnica de aprendizagem supervisionada e uma estrutura de rede neural *feedforward* multicamadas na qual vários neurônios perceptron são empilhados em camadas.

- Uma camada de entrada;

- Uma ou mais camada oculta;
- E uma camada de saída.

O número de camadas e o número de neurônios em cada camada são chamados de hiperparâmetros da rede neural, que são predefinidos pelo desenvolvedor com base na experiência. Cada neurônio tem um peso w para uma entrada, um viés b que é chamado de bias e uma função de ativação. A regra de ativação do perceptron que é seguida pelo MLP é muito simples e está na base de muitas redes neurais modernas.

Dada uma quantidade de entradas que é tipicamente um vector de características $x = (x_1, \ldots, x_n)$, a soma ponderada de cada dimensão da entrada x_i e seu peso associado w_i é calculado como:

$$z = \sum_{i=1}^{n} (w_i \cdot x_i) + b,$$

É um processo de otimização para encontrar a convergência enquanto altera os valores de w e b para descobrir o valor de y desejado. Neste sentido, o MLP aprende a encontrar a melhor aproximação de uma função $\sigma(z)$ para esse classificador.

Por exemplo, dado um classificador $y = f(z) * x$ que mapeia uma entrada x para uma classe de saída y. Uma rede com três funções ou camadas forma $f(x) = f(3)(f(2)(f(1)(x)))$. Cada uma dessas camadas consiste em unidades que realizam uma transformação através da soma linear das entradas. Cada camada é denotada por $y = f(w * x + b)$ onde f é a função de ativação, w é o conjunto de parâmetros ou pesos da camada, x é o vetor de entrada que pode ser a saída da camada anterior e b é o vetor de bias. Esses parâmetros (pesos, bias) são exatamente os valores numéricos que tentaremos ajustar treinando a rede com um conjunto de dados já rotulado, como em qualquer outro problema de ML supervisionado.

A Figura 3.9 ilustra a estrutura de seu funcionamento, onde o algoritmo busca otimizar os pesos w_i.

O treinamento da rede neural geralmente tem que passar por muitas rodadas, cada rodada tem um processo de propagação para frente e para trás, e os parâmetros são ajustados em combinação com o gradiente descendente (Popescu et al., 2009) que é um cálculo numérico que nos permite saber como ajustar os parâmetros de uma rede de forma que seu desvio de saída seja minimizado. Mas o gradiente é obtido com o algoritmo de *retropropagação* que é geralmente usado para treinar o perceptron multicamadas MLP. Usar o algoritmo de retro-propagação é como aprender com os erros. Ele corrige quando a rede neural artificial comete um erro.

O objetivo do aprendizado é atribuir os pesos corretos a essas arestas. Por meio do vetor de entrada, esses pesos podem determinar o vetor de saída. Na aprendizagem supervisionada, o conjunto de treinamento é rotulado. Isso significa que, para alguma entrada fornecida, a saída esperada (rótulo) pode ser conhecida.

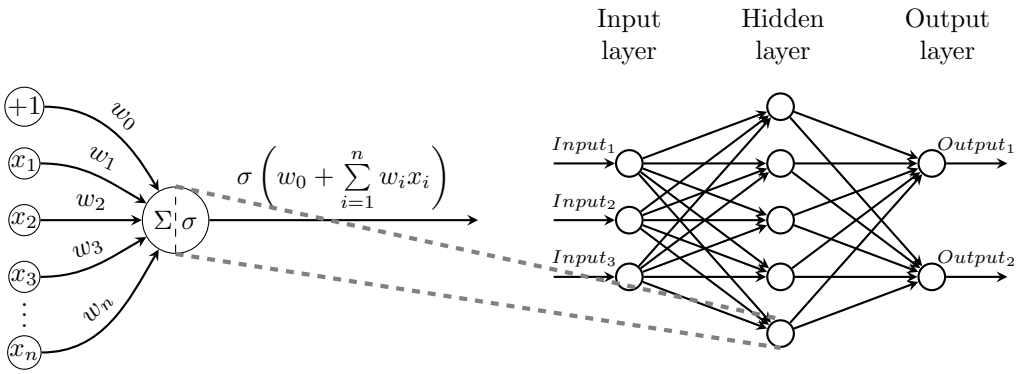


Figura 3.9: Representação da estrutura de um único perceptron, junto com sua posição dentro de um perceptron multicamadas em grande escala. Fonte: Adaptado pelo autor https://github.com/PetarV-/TikZ/blob/master

3.9.1 Estrutura do aprendizado da MLP

O processo de treinamento da MLP é realizado da seguinte forma:

(a) Os pesos de todas as arestas da entrada são atribuídos aleatoriamente;

(b) Propagação direta: Usando as característica de entrada de todas as amostras no conjunto de treinamento como a camada de entrada, para todas as entradas no conjunto de dados de treinamento, a rede neural artificial é ativada e, em seguida, o valor de saída é obtido por meio da propagação direta.

(c) Retro-propagação: Com a descida gradiente para minimizar os erros que é calculado na diferença entre os valores de saída das redes e os rótulos corretos de cada instância da base de treinamento. Em seguida, é usada a retro-propagação para atualizar o peso.

(d) Repita (b) e (c) até que o erro de saída seja inferior ao padrão estabelecido.

O algoritmo de retropropagação calcula o gradiente da função de perda para o valor de entrada de cada neurônio enquanto flui o gradiente de erro da camada de saída para a camada de entrada. O gradiente calculado desta forma é aplicado a cada parâmetro de peso w_i e é atualizado.

3.10 Redes Neurais Convolucionais (CNN)

Em redes neurais multicamadas, é comum esperar que a capacidade de aprendizado melhore conforme o número de camadas ocultas aumenta. Mas, na prática, à medida que o número de camadas aumenta , apesar da forte capacidade de adaptação das redes neurais, depois de muitos treinamentos, pode ocorrer problemas como desaparecimento do gradiente e *overfitting*. O erro pode gradualmente desaparece convergindo para 0 conforme a retropropagação passa por cada uma das camada. Outra desvantagem é a quantidade aumentada de computação, o que sobrecarrega o hardware, resultando em um treinamento muito lento. Esses problemas foram posteriormente melhorados de várias maneiras. O problema de perda de gradiente pode ser resolvido aplicando *ReLU* (*Rectified Linear Unit*) à função de ativação.

Para evitar *overfitting* pode ser solucionado com técnicas *Dropout* que refere-se ao uso de meios artificiais para desligar alguns nós das redes neurais, o que torna esses nós inválidos (Cheridito et al., 2021).

Quanto ao tempo de aprendizado lento, (GPU, CUDA) foi melhorado rapidamente devido ao desenvolvimento de placas gráficas.

Consequentemente, foram desenvolvidas vários tipos de redes neurais e uma derivada delas é a rede neural convolucional CNN que é uma estrutura de rede neural artificial otimizada de estrutura inspirada no processo de percepção visual humana que utiliza a rede neural *feedforward* para a classificação. No método convencional das RNAs, o aprendizado é feito extraindo conhecimento dos dados recebidos, em contraposição, a CNN é uma estrutura que extrai características como recurso e apreende os padrões dessas características. Seu algoritmo prossegue usando a camada de convolução, camada de *pooling* (subamostragem) e camada totalmente conectada que é denominada redes neurais *feedforward*. A camada de convolução é usada para encontrar os pontos característicos da imagem, a camada de *pooling* é usada para reduzir o peso e a computação necessária para o processamento da imagem e a camada totalmente conectada é usada para classificar a imagem. A chave para a CNN é quebrar toda a imagem em pequenas unidades e analisar cada parte e tem a vantagem pela pequena quantidade de computação e alto desempenho em comparação com a MLP existente.

3.10.1 Estrutura do aprendizado da CNN

Primeiramente, como nos modelos do perceptron, há a camada de entrada. Uma CNN pode aceitar matrizes 1D, 2D ou 3D como entrada, tornando-as perfeitas para imagens e dados de matrizes mais complexas.

Depois de definir as entradas, a CNN consiste em dois tipos de camadas ocultas para realizar cálculos e a camada *feedforward* para fazer a classificação.

- A primeira delas é uma camada de convolução que divide a imagem em pedaços menores e aplica vários filtros para ver se há um atributo específico dentro do pedaço da imagem. Na parte inicial da rede neural, pode ser um filtro que determina bordas retas horizontais ou retas verticais, e os filtros na parte inferior da rede podem encontrar círculos (que lembram o olho de um animal) ou padrões em forma de cruz. O filtro a ser usado aqui é determinado pelo processo de treinamento. Isso significa que o valor da matriz do filtro é determinado por meio do aprendizado. Os mapas de características obtidos por convolução ou filtragem são produzidos por meio de funções de ativação, como *ReLU*.

- Ao receber os atributos da imagem derivados da camada convolucional implementada anteriormente, constatar-se resultados ainda grande e complexo, precisando reduzi-lo novamente. Este processo é denominado *pooling* ou subamostragem. Técnicas de *pooling* como; *Max polling* (pega o maior valor), *Average pooling* (pega a média) são usadas neste processo conforme a necessidade da técnica escolhida. Estes valores são passados para a próxima camada e o restante é descartado.

- O mapa reduzido de características é inserido na rede neural *feedforward* para classificar aplicando uma função de ativação *softmax.*

Em resumo, pode-se dizer que a CNN possui uma estrutura na qual uma rede neural de extração de características e uma rede neural de classificação são conectadas em série. Isto fica claro na Figura 3.10.

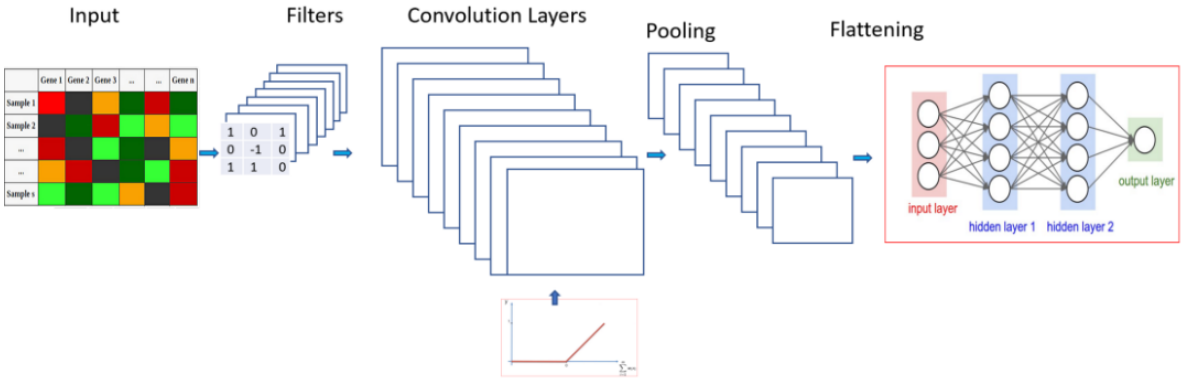


Figura 3.10: Convolução, ativação e subamostragem são realizadas iterativamente nos mapas de características para criar gradualmente um mapa de atributo global. O mapa de atributo final obtido pela repetição desse processo várias vezes é inserido em uma camada totalmente conectada, ou seja, uma rede neural *feedforward* para realizar tarefas de classificação. Fonte: Retirado de https://medium.datadriveninvestor.com/convolutional-neural-network-cnn-simplified-ecafd4ee52c5 e adaptada pelo autor.

3.11 Pré-processamento de Dados

É comum que problemas de classificações apresentem diversas características que são desnecessárias ou que não deveriam ser desconsideradas pelos classificadores, podendo prejudicar o modelo de classificação criados pelos mesmos.

Com isso, diversas técnicas de seleção e transformação de características foram propostas de forma que possam ser utilizadas como pré-processamento dos dados com o objetivo de melhorar a estabilidade, a velocidade de convergência e o desempenho dos classificadores.

3.11.1 *Normalização*

Normalmente, os dados podem precisar de um redimensionamento quando a escala de um número varia muito. Quando o valor dos dados é muito grande ou muito pequeno, ele evita a convergência para 0 ou diverge para o infinito durante o processo de treinamento ou ainda, impede que o valor de predição seja mais influenciado pelo valor dos dados de entrada fora do intervalo. Por exemplo, se a altura e o peso são informações como dados de entrada e os dados do rótulo preveem o tempo que leva para correr 100 m, a altura e o peso têm faixas e unidades diferentes, então o valor mais alto tem um efeito maior no resultado. Para evitar isso , é preciso trabalhar para combinar as unidades dos dados de altura e peso. Nesse caso, *Mini_Max_Scaler* do *Scikit-Learn* é usado neste trabalho, tornando a entrada muito mais adequadas para os algoritmos que queremos empregar.

3.11.2 *Análise de Componentes Principais (PCA)*

A Análise de Componentes Principais PCA tem por objetivo a redução da dimensionalidade do vetor sem perder as informações utilizadas para a classificação. Pertence a um método comumente utilizado na análise estatística multivariada, que extrai efetivamente os parâmetros que podem fornecer as principais informações para a classificação por meio da transformação da matriz, de forma a facilitar a análise. Ideal para problemas onde o número de característica é

muito maior do que o número de exemplos. Esta é a razão da escolha deste pré-processamento, já que as bases de dados em questão deste trabalho possuem esta característica.

3.12 Validação Cruzada (CV)

A validação cruzada estima o erro e o desempenho de generalização de um modelo de modo imparcial e pode reduzir possíveis viés. Na construção do modelo, geralmente há dois conjuntos de dados: conjunto de treinamento e conjunto de teste . O conjunto de treinamento é usado para treinar o modelo; o conjunto de teste são dados que não participam de nenhum treinamento e são usados apenas para observar o efeito do teste. Com o tradicional método *Holdout* os dados são divididos aleatoriamente para treino e teste. Ou seja, os dados que antes eram utilizados para teste agora podem estar sendo utilizados para treino e vice-versa. Para evitar a aleatoriedade dos dados, a CV busca estimar a performance do modelo em dados que nunca viu antes e buscar também, um desempenho de generalização imparcial.

3.12.1 *Validação Cruzada Estratificada*

Alguns conjuntos de dados não são distribuídos uniformemente, assim sendo, haverá grandes desequilíbrios após o treinamento do modelo. Com isto, é necessária uma amostragem estratificada, ou seja, considera a distribuição das classes, de modo que a distribuição de cada partição de treinamento ou validação dependa da distribuição de todo conjunto de dados.

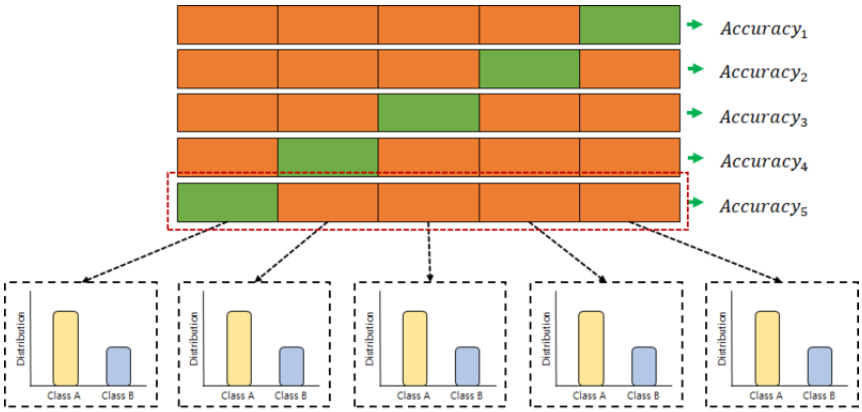


Figura 3.11: Distribuição balanceadas das classes nas partições da CV *Kfold* estratificada. Fonte: (Zeng e Martinez, 2000)

3.12.2 *Nested Cross-validation*

A validação cruzada aninhada (*Nested Cross-validation*) costuma ser usada para treinar um modelo no qual os hiperparâmetros também precisam ser otimizados. Este método escolhe o melhor modelo em uma lista de modelos candidatos e suas listas de parâmetros associadas. Em vez de fazer uma única divisão de de treino e teste, deve-se usar uma segunda divisão de validação cruzada. Ou seja, "aninha-se"uma divisão da validação cruzada "interna"dentro de uma divisão de validação cruzada "externa". A divisão interna da validação cruzada é usada para escolher hiper-parâmetros. O divisão externa de validação cruzada calcula a média do

erro de teste em várias divisões de teste e treino. A média do erro de generalização em várias divisões fornece uma estimativa mais confiável da precisão do modelo em dados não vistos. Em suma, a parte externa do *loop* da validação cruzada aninhada é dividida em *Kfold* , e o *loop* interno executa a validação cruzada para selecionar um modelo com hiperparâmetros. Quando um modelo é selecionado, ele é avaliado como uma partição de teste.

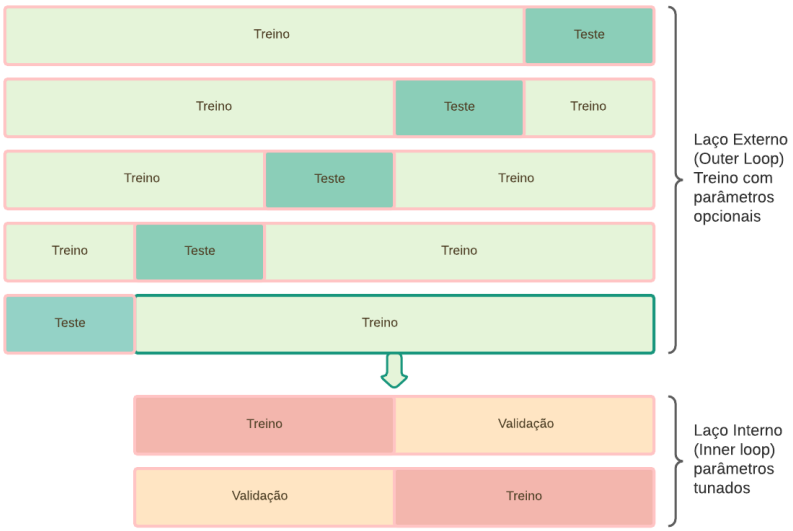


Figura 3.12: Exemplo de CV aninhada (aqui 5x2 CV aninhada), ou seja no *loop* externo a divisão é em 5 *Kfold* e *loop* interno 2 *Kfold*. Fonte: Desenho inspirado em https://www.programmersought.com/article/66124607365/

3.13 Métricas de Avaliação

Nesta seção, descrevemos as métricas de desempenho que são usadas neste experimentos.

- **Verdadeiros positivos (TP)**: os casos em que SIM foi previsto e a saída real também foi SIM.
- **Verdadeiros Negativos (TN**: os casos em que NÃO foi previsto e a saída real foi NÃO.
- **Falsos Positivos (FP)**: Os casos em que SIM foi previsto e o resultado real foi NÃO.
- **Falsos negativos (FN)**: os casos em que NÃO foi previsto e a saída real foi SIM.
- *Matriz de confusão*: É fácil usar uma matriz de confusão para visualizar os resultados de um classificador. Podemos calcular as métricas mencionadas acima como: TP, FP, TN, FN que constrói uma tabela 2x2 formulada com esses quatro resultados.
- *Accuracy*: É a razão entre o número de previsões corretas e o número total de amostras de entrada.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- *Precision*: É a razão entre o número de resultados positivos corretos e o número de resultados positivos previstos pelo classificador. Quando o custo do Falso Positivo (PF) é muito alto (as consequências são muito graves), ou seja, quando se deseja evitar ao máximo o PF, devemos focar na melhoria do índice de Precisão.

$$Precision = \frac{TP}{TP + FP}$$

- *Recall ou Revocação ou taxa de positivo verdadeiro*: É a razão entre o número de resultados positivos corretos e o número de todas as amostras relevantes (todas as amostras que foram identificados como positivos). Quando o custo do Falso Negativo (FN) é muito alto (as consequências são muito graves) e espera-se evitar a geração de FN tanto quanto possível, devemos nos concentrar em melhorar o indicador de *recall*.

$$Recall = \frac{TP}{TP + FN}$$

- *F1_score*: A pontuação F1 é útil quando você deseja buscar um equilíbrio entre precisão e revocação. A média harmônica da precisão e *recall*. É um indicador usado para medir a precisão de dados não balanceados. O método de cálculo de precisão é a média ponderada da precisão de cada classe e seu peso. O *recall* é o mesmo, e F1 é calculado diretamente por precisão e *recall*.

$$f1\_score = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{recall}}$$

- *Taxa de falso positivo (FPR)*: É a proporção de observações identificadas incorretamente.

$$FPR = \frac{FP}{FP + TN}$$

- *Taxa de alarmes falsos (FAR)*: A probabilidade de um registro ser classificado incorretamente.

$$FAR = \frac{FP + FN}{FP + FN + TP + TN}$$

- *ROC AUC*:

  A curva ROC, é um gráfico que ilustra o desempenho da Curva Característica de Operação do Receptor que é um sistema classificador binário conforme suas configurações de limite. A AUC (Area Under Curve) é definida como a área sob a curva ROC e representa o valor da probabilidade de um exemplo aleatório, por exemplo, positivo (doente) obtenha uma pontuação maior de um exemplo aleatório negativo (não doente) (Zohuri et al., 2021). Quando se seleciona aleatoriamente uma amostra positiva e uma amostra negativa, o algoritmo de classificação atual classifica a amostra positiva antes da amostra negativa de acordo com o valor de pontuação calculado (Powers, 2020). Quanto maior o valor AUC , o atual algoritmo de classificação tem maior probabilidade de classificar as amostras positivas antes das amostras negativas, para que possa classificar melhor. Obviamente, o valor desta área não será maior que 1. E quanto mais a curva se aproxima do canto superior esquerdo melhor será a precisão da classificação. A curva ROC é desenhada por duas variáveis indicadoras: especificidade e sensibilidade.

A especificidade FPR, ou seja, (Taxa de classe falso positivo). A sensibilidade é a taxa de classe verdadeira, TPR (Taxa de classe positiva verdadeira), que reflete o grau de cobertura de classe positiva. Essa combinação é baseada na especificidade versus sensibilidade, ou seja, custos versus benefícios.

$$Specificity = FPR = \frac{FP}{FP + TN}$$

$$Sensitivity = Recall = TPR = \frac{TP}{TP + FN}$$

3.14 Dados de Microarray

O objetivo do projeto Genoma Humano, que chamou a atenção do mundo, era decifrar a sequência de nucleotídeos dos genes contidos nos 22 pares de autossomos dos humanos e dos cromossomos sexuais representados por X e Y e criar um mapa do genoma. Os pares de bases que compõem o Genoma Humano foi sequenciado e possui cerca de 30.000 a 40.000 genes, ou seja, regiões codificadoras do DNA que são importantes na montagem de proteínas ou regulação de outras regiões (Walker, 2002). Como resultado de uma pesquisa bem-sucedida, o mapa do genoma humano foi concluído em fevereiro de 2002 (Walker, 2002). Este é um mapa 99% detalhado (Liang et al., 2000) que mostra a localização de genes específicos por meio de marcadores biológicos. Com a conclusão do mapa, abre-se a era pós-genoma, que revela as funções biológicas de cada gene do genoma humano e compara as informações do genoma entre organismos para identificar diferenças nas funções biológicas. Como o projeto do genoma humano revelou ser de uma vasta quantidade de dados e com as informações de sequência genética completa de vários indivíduos, surge a questão de como interpretar e usar a vasta quantidade de dados obtidos, tornando-se um problema mais difícil e imediato. O método convencional de pesquisa baseada em biologia molecular consiste em experimentos de um único gene a sua observação é limitada. Em consequência disto, a tecnologia de *microarray* de DNA foi desenvolvida e tem a vantagem de pesquisar o padrão de expressão de todo o genoma em um único chip e ao mesmo tempo, observar a interação entre milhares de genes, ou seja, pode detectar a atividade de centenas ou mesmo milhares de genes em paralelo. Além disso, tem potencial para produzir uma grande quantidade de dados em um curto espaço de tempo.

Os *microarrays* são baseados principalmente em matrizes bidimensionais, como vidro ou lâmina de silício, permitindo medir biomoléculas qualitativamente ou quantitativamente (como DNA, ácido ribonucleico RNA, proteína, etc.).

A estrutura das fitas opostas de DNA é formada por pares de bases complementares através de ligações de hidrogênio. Estas bases complementares são sequências de ácidos nucleicos e se apresentam de quatro tipos: adenina (A), citosina (C), timina (T), e guanina (G). A complementaridade das bases é denominada de hibridização molecular entre duas fitas de DNA e é o princípio básico da técnica dos *microarrays*, obtendo assim, informações sobre a estrutura e expressão do gene. Sua origem técnica foi desenvolvida em 1975 pelo Dr. Edward Southern da Universidade de Edimburgo (Southern, 2001), observou que uma molécula de ácido nucleico marcada poderia emparelhar com outra molécula de ácido nucleico imobilizada, usando a técnica de hibridização para a determinação da sequência. As tecnologias como clonagem dos anos 80, PCR dos anos 90, são desenvolvidas com base nas mesmas características de ligação

complementar dos ácidos nucleicos e ainda hoje, os métodos *microarrays* são muito utilizados e uns dos mais valiosos no campo das ciências biológicas.

Processo de experimento de *microarray*:

(a) *Coleta de amostra* - Colete células normais (normais) e células cancerosas (doenças) para verificar a expressão do gene.

(b) *Isolamento de RNA mensageiro (mRNA)*

- Usar um solvente orgânicos como fenol-clorofórmio que tem como função separar e para extrair o RNA das proteínas.
- Remover RNA ribossômico (rRNA) e RNA transportador (tRNA) do RNA extraído, deixando apenas (mRNA).

(c) *Geração de cDNA rotulado*

- Realizar a transcrição reversa de mRNA para gerar DNA complementar (cDNA).
- Use diferentes marcadores fluorescentes para distinguir o normal da doença.

(d) *Hibridização*

- Os cDNAs sintetizados são espalhados no chip para iniciar a hibridização.
- Sequências não ligadas são removidas por lavagem.

(e) *Análise de instrumento*

- Gravar imagens de fluorescência de cDNA com *scanner* de *microarray*
- De acordo com a diferença na intensidade de fluorescência para cada ponto, o nível de expressão do gene é medido:
  * Verde: Normal
  * Vermelho: Câncer
  * Amarelo: Comum

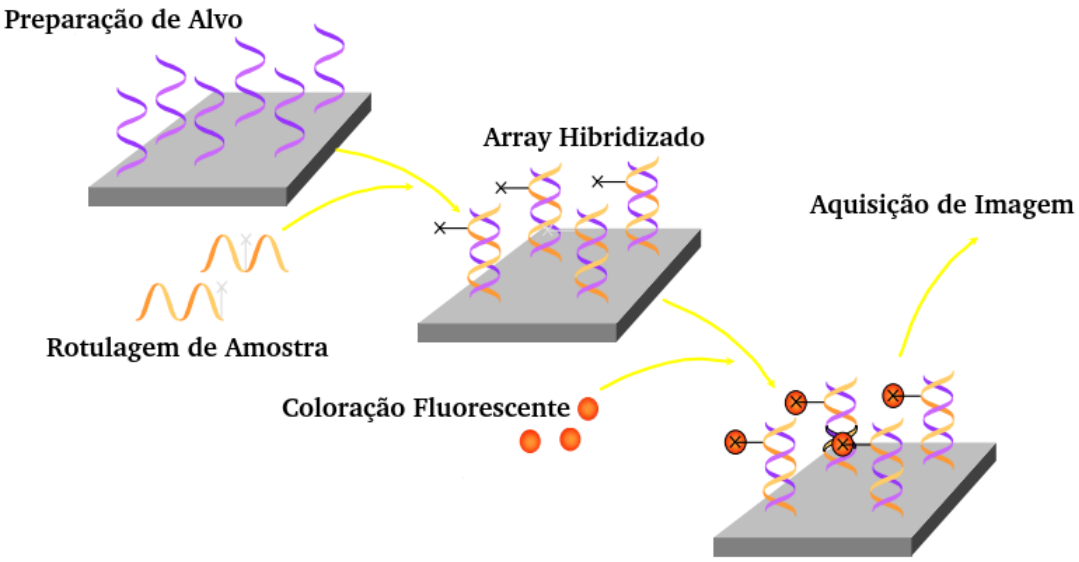


Figura 3.13: Etapas envolvidas no experimento de *microarray*.
Fonte: Retirado de
https://www.thermofisher.com/us/en/home/life-science/microarray-analysis.html

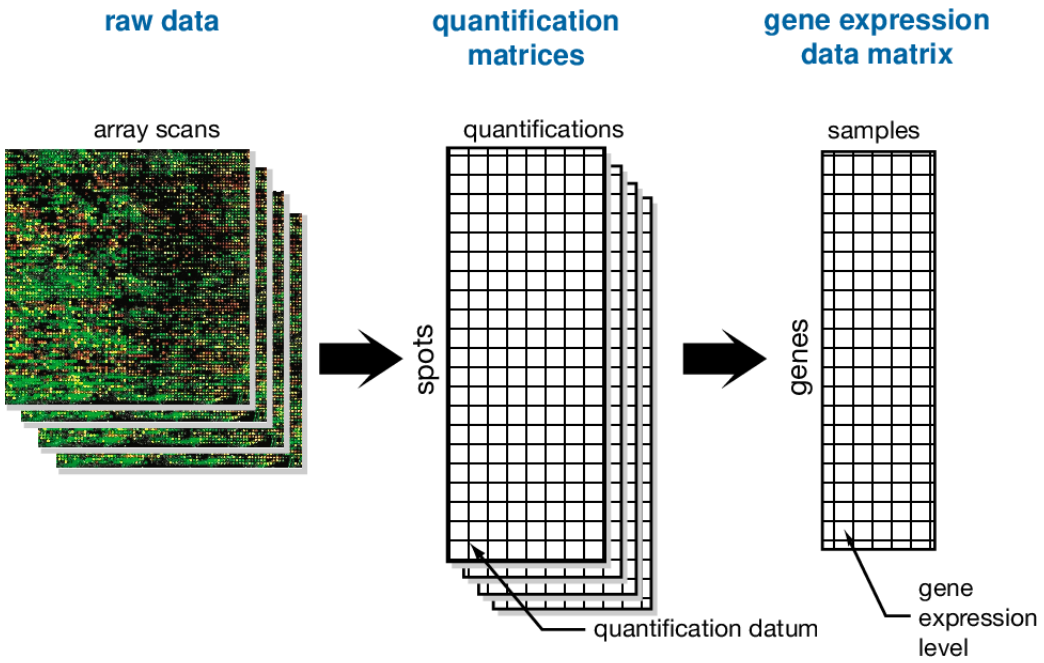


Figura 3.14: Três níveis de processamento de dados de expressão gênica de*microarray*. Fonte: (Bolón-Canedo et al., 2014)

3.15 Análise de Expressão Gênica

O processamento de dados de expressão gênica de *microarray* é uma área multidisciplinar da ciência da computação que abrange a área de ML, *clustering* e classificação (Djebbari et al., 2011). Tecnologia de *microarray* permite medir vários milhares de dados de expressões gênicas em um único experimento. Níveis de expressão gênica ajudam a determinar genes correlacionados e progressão da doença, o que por sua vez ajuda no diagnóstico precoce e prognóstico de diferentes tipos de câncer.

Em um organismo, qualquer célula carrega a mesma informação genética e os mesmos genes, mas o desempenho de um gene em diferentes tecidos e células é diferente, o que é determinado pelo mecanismo de regulação gênica. O processo de transferência de informações genéticas do DNA para a proteína é chamado de expressão gênica, e a regulação desse processo é a regulação da expressão gênica. Em um determinado momento, uma célula expressa apenas uma pequena parte do genoma. O mecanismo de regulação gênica controla com precisão a quantidade de produção de cada proteína de acordo com os requisitos funcionais de cada célula. O processo de vida completo de um organismo é o resultado de cada gene no genoma sendo ativado e desativado em uma certa ordem.

De acordo com as mudanças nas condições ambientais, os organismos sintetizam várias proteínas para adaptar seus processos metabólicos às mudanças no ambiente.

A diferenciação celular ocorre durante o desenvolvimento individual e forma vários tecidos e órgãos. As proteínas sintetizadas por diferentes tipos de células são diferentes em qualidade e quantidade. Portanto, as células procariotas e eucariotas têm um conjunto de mecanismos de regulação precisos para a expressão gênica e síntese de proteínas (Alberts et al., 2002). E para que as células mantenham suas funções, algumas proteínas são necessárias o tempo todo. Os genes correspondentes a essas proteínas são chamados de genes de manutenção e devem ser expressos a qualquer momento. Os genes que codificam proteínas específicas de células são chamados de genes induzidos, e esses genes são expressos apenas quando e onde a proteína

correspondente é necessária. Embora cada célula de um organismo tenha um genoma completo, a expressão de vários genes em células diferentes é diferente. Neste ponto podemos discernir a importância da ajuda de ferramentas de ML para identificar possíveis alterações malignas na forma de como o gene se expressa. Para entender as leis de crescimento e desenvolvimento, características morfológicas e estruturais e funções biológicas dos organismos, é necessário estudar as leis de tempo e espaço da regulação da expressão gênica e dominar os segredos da regulação da expressão gênica (Brazma e Vilo, 2000).

A expressão do gene dispõe de diferentes etapas que podem ser moduladas, incluindo a transcrição , *splicing* de RNA , tradução e modificação pós-tradução do produto gênico sintetizado que são as proteínas.

Isso significa que os genes são expressos ao serem transcritos em RNA , e esse transcrito pode então ser traduzido em proteína. Portanto, é necessário se obter o RNA para poder avaliar a expressão de um determinado gene, então se obtém a transformação do RNA em DNA (cDNA) para uma precisa análise da expressão gênica.

Este processo de transformação é chamado de *Dogma Central da Biologia Molecular* como é visto na Figura 3.15 que descreve o processo de duas etapas, transcrição e tradução, pelo qual a informação nos genes flui para as proteínas:

DNA → RNA → proteína.

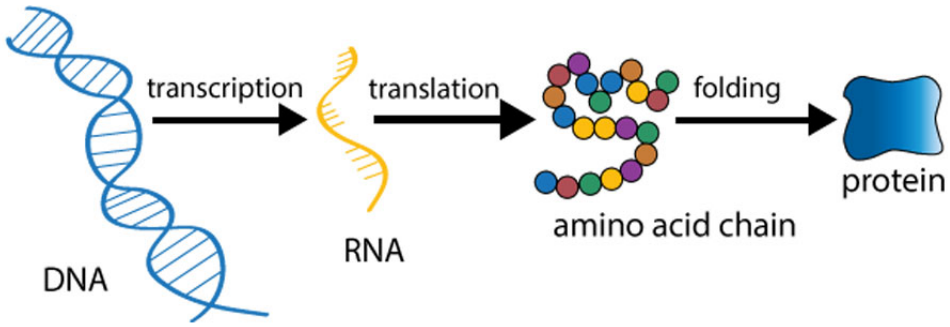


Figura 3.15: Etapas modulares da expressão gênica incluindo a transcrição, *splicing* de RNA, tradução e modificação pós-tradução de um determinado produto gênico sintetizado (proteína). Fonte: (Morange, 2009)

4 Revisão Bibliográfica

4.1 Processo de Aprendizagem em Pesquisa de Bioinformática

Nas últimas décadas, ao analisar e quantificar variáveis moleculares, como genes, proteína e variação genética populacional, o campo da biologia produziu um grande número de conjuntos de dados altamente complexos. Esses conjuntos de dados fornecem as matérias-primas necessárias para uma compreensão aprofundada de sistemas biológicos e doenças complexas. Mas para explorar ainda mais o valor desses dados, precisaria ser realizado por meio de um processo de análise de nível superior. E como o número de conjuntos de dados está aumentando rapidamente, a pesquisa interdisciplinar frequentemente produz uma enormidade de petabytes de dados. E ainda, os sistemas de IA estão se tornando mais comuns nos laboratórios e clínicas para dar suporte no sentido do acúmulo crescente de dados genômicos. De acordo com o relatório divulgado por (Pauwels e Vidyarthi, 2017), estima-se que até 2025, o mundo vai gerar um total de 1 bilhão de dados do genoma completo, e o custo da detecção de genes cairá para quase a 0. Considerando que todos os dados do genoma de todos chegarão a uma quantidade massiva, como interpretar esses dados? Ferramentas e tecnologias para processar grandes conjuntos de dados heterogêneos e complexos estão se tornando extremamente importantes. Esta é uma oportunidade para a inserção de ML e DL em bioinformática, um campo de pesquisa que lida com grandes conjuntos de dados multidimensionais.

4.2 Predição de Dados de Expressão Gênica no Estado da Arte

A análise bioinformática usando informações de expressão gênica tem sido estudada com os algoritmos de ML e DL. Em vários estudos desses algoritmos provou ser uma técnica robusta para analisar conjuntos de dados em grande escala (Bengio et al., 2013). Com o avanço do DL essas práticas alcançaram um desempenho de ponta em uma ampla gama de aplicações na bioinformática e genômica (Min et al., 2017), onde o sequenciamento do genoma capacitou os pesquisadores a buscar novos desafios, sendo assim, estão se esforçando para lutar contra várias doenças genéticas, como o câncer. A fim de identificar dados importantes para eventos adversos graves ou previsão de resultados. (Montesinos-López et al., 2019) propôs modelos de DL de múltiplas características disponíveis para a previsão simultânea de fenótipos moleculares, como locais de ligação de fatores de transcrição, diferentes marcadores de histonas, acessibilidade ao DNA e expressão gênica em diferentes tecidos.

Tais metodologias podem selecionar atributos que caracterizam classes, identificam grupos com espaço de característica semelhantes, classificar casos ou dados mistos. Esses métodos têm sido aplicados no contexto da classificação genômica multinível, especialmente para a pesquisa do câncer (Ibrahim et al., 2014).

DeepChrome propôs (Singh et al., 2016) uma estrutura unificada que coloca uma camada de MLP no topo da CNN, para modelar a regulação gênica. Foi a primeira abordagem de DL para prever o nível de expressão do gene e capturou características locais de marcas de histonas.

AttentiveChrome (Singh et al., 2017) aprende a prever a expressão de um gene a partir de uma entrada de sinais de modificação de histonas cobrindo a região vizinha de DNA do gene. Esta abordagem usa uma hierarquia de vários módulos de memória de longo prazo com um mecanismo de atenção para prever os níveis de expressão gênica. Este modelo pode descobrir as interações entre os sinais de cada marca da cromatina e, simultaneamente, aprender dependências complexas entre as diferentes marcas. Este modelo previu a expressão gênica com mais precisão do que o *DeepChrome* que já tinha sido um avanço. Ambos os estudos usaram abordagens individuais de DL para entender a regulação gênica, mas utilizaram apenas marcas de histonas. Outras informações epigenéticas, como metilação do DNA, miRNA (Kuhn et al., 2008), características da cromatina, etc. também podem ser usadas em tarefas de predição de expressão gênica. Uma interessante solução para classes desbalanceadas foi proposta por (Al-Stouhi e Reddy, 2016) chamada aprendizagem por transferência que no caso de dados escassos, pode não ser viável treinar o modelo do zero. Em vez disso, a maioria dos parâmetros de outro modelo treinado em uma tarefa semelhante podem ser usados para inicializar o modelo. Este método pode ser visto como uma integração do conhecimento prévio ao modelo.

E na literatura diversos trabalhos abordam problemas de classificação com a análise de expressão gênica por meio do processamento de dados provindos de *microarrays* DNA, onde frequentemente se utiliza os algoritmos de ML e DL. No entanto, os dados de *microarrays* contêm a expressão de milhares de genes, embora haja um número limitado de amostras disponíveis para análise.

Este modelo de conjunto de dados é um bom exemplo da alta dimensionalidade. Devido a este aspecto, este problema se apresenta ser desafiador para a previsão de classe, pois muitas vezes resulta em um alto erro de generalização.

Uma solução eficaz para aliviar o problema é realizar a seleção de genes para reduzir a dimensionalidade dos dados de *microarrays* como nos exemplos de (Wong e Chen, 2011), (Nakariyakul, 2016), (Guyon et al., 2008), (Tan et al., 2014) e classificação de câncer usando PCA (Liu et al., 2011), além do mais técnicas de pré-processamento como: (Bolón-Canedo et al., 2014); (Wang e Wei, 2017); (Perera et al., 2018); (Liu et al., 2019) , entre outras. Portanto, a maioria dos estudos usa *microarrays* para testar técnicas de ciência de dados específicas, como métodos de seleção de atributos.

Em estudos anteriores demonstram o uso de ML e DL na expressão de genes de *microarray* para inferir a expressão de genes alvo com base na expressão de genes de referência (Chen et al., 2016), na seleção de atributos com o objetivo de encontrar um subconjunto informativo de expressão de genes (Fakoor et al., 2013) , e no diagnóstico e classificação dos tipos de câncer. (Golub et al., 1999) descrevem uma abordagem para a classificação do câncer com base na expressão gênica aplicada à leucemia mieloide aguda (LMA) e leucemia linfoblástica aguda (LLA), ele mostrou como novos casos de câncer podem ser classificados pelo monitoramento da expressão gênica (via *microarray de DNA* e, assim, forneceu uma abordagem geral para identificar novas classes de câncer e atribuir tumores a classes conhecidas e (Alizadeh et al., 2000) desenvolveu (*Alive vs Dead*) um modelo preditor de sobrevivência do paciente após a quimioterapia.

Em (GÜÇKIRAN et al., 2019) que usa alguns algoritmos de ML para classificação como SVM, MLP e RF que são amplamente usados em várias tarefas de classificação, similar a este atual trabalho que usa os mesmos algoritmos e algumas das bases de dados como: (Gordon et al., 2002), (Khan et al., 2001), (Su et al., 2001), (Yeoh et al., 2002) usadas neste experimento.

Além disso, o artigo usa diferentes técnicas de pré-processamento nas bases com os métodos de seleção de atributos como Relief (Kira e Rendell, 1992) e Lasso (Tibshirani, 1996), e conseguiram ótimos resultados com precisão de 100% para alguns algoritmos como o SVM.

Algumas das bases usadas neste trabalho possuem a quantidade de exemplos para cada classes desbalanceadas e umas das soluções apresentadas em estudos anteriores é realizar a CV estratificada como em (Parker et al., 2007) e recomenda a utilização em dados de *microarrays*.

No artigo de (Tabares-Soto et al., 2020) faz uma comparação dos métodos ML e DL na classificação de 11 classes diferentes de 11_Tumores (Su et al., 2001) usando dados de expressão gênica de *microarray* a mesma usada neste trabalho. Seus resultados mostraram que os métodos de DL superaram os outros métodos de ML. Trabalhos recentes como o de Guillen e Ebalunode (2016) demonstraram resultados promissores para a aplicação do MLP como a mesma base de *microarrays* 11_Tumores (Su et al., 2001). Mas com a diferença na implementação, a biblioteca H20 foi usada como abordagem para classificar com eficiência o câncer de rim, câncer de pulmão e câncer de fígado.

Outra questão encontrada pelos pesquisados é de como evitar o *overfitting*. Eles discutem os modelos tradicionais de CV que pode ter um viés nos dados de treinamento e podem fazer uma seleção tendenciosa e não realista, tornando-o não confiável para muitas aplicações. E uma das proposta feita por (Cawley e Talbot, 2010) e (Varoquaux et al., 2017) para este problema foi aplicar a CV aninhada que podem ser usadas tanto na seleção e na avaliação do modelo. E ainda, uma CV simples, produz uma estimativa imparcial, enquanto que a CV aninhada produz várias estimativas imparciais.

Em uma visão geral sobre o que já vimos sobre o desempenho das aplicações é de que os métodos DL mostraram lidar com melhor eficiência em dados de larga escala. Visto que as camadas das redes neurais tem o poder de filtrar e extrair as características. Essas qualidades de DL atraem pesquisadores biomédicos para usar DL em vez de métodos de ML convencionais porque os dados biomédicos por exemplo, dados *ômicos* muitas vezes sofrem com a alta dimensionalidade e também, podem conter dados ruidosos.

5 Materiais e métodos

Esta seção discute sobre os conjuntos de dados e metodologia adotada neste trabalho.

5.1 *Bases de Dados*

Neste experimento, conjuntos de dados de expressão gênica são usados para examinar o desempenho de classificação do método proposto. Eles representam medições de expressão gênica retratada pelos *microarrays* de câncer e biópsias normais das suas propriedade como:

- tipos de câncer;
- tipo de matriz (que compreende as duas plataformas em que a tecnologia está geralmente disponível), ou seja:
  - de canal único *Affymetrix (Irizarry et al., 2003);*
  - *de canal duplo cDNA (DeRisi et al., 1996)*;
- número de amostras;
- número de classes;
- amostras por classes e
- número de atributos.

Para cada base de dados de *microarrays* que por sua vez refletem uma matriz de dados de expressão gênica onde se organiza em *n* colunas de genes e *m* linhas de amostras. A última coluna é o rótulo da classe, ou seja, informações sobre qual é a classe da amostra. Um exemplo da configuração da matriz de expressão gênica usada neste experimento, pode ser vista na Figura 5.1.

Nome da Amostra	*Gene 1*	*Gene 2*	*Gene 3*	*Gene 4*	*Gene 5*	*Gene n*	*Classes*
Amostra 1	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{1n}	$y1$
Amostra 2	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{2n}	$y2$
Amostra 3	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{3n}	$y3$
Amostra m	x_{m1}	x_{m2}	x_{m3}	x_{m4}	x_{m5}	x_{mn}	ym

Figura 5.1: Matriz de tamanho $m \times n$ e as classes são um vetor de tamanho $m \times 1$, onde m é o número de amostras e n é o número de genes para cada classe. As características compõem a representação por algum valor de x e as classes pelos valores de y. Fonte: Elaborada pelo autor e inspirada em (Mandal e Mukhopadhyay, 2013)

Estes *microarrays* de expressão gênica são geralmente simbolizados por inteiros e reais com valores positivos e negativos. Esses valores retratam as características indicadas pelo valor de x na Figura 5.1 que permitem aos algoritmos de ML e DL aprender a classificar por tipo de câncer. E normalmente uma propriedade dos *microarrays* é de que muitos conjuntos de dados de expressão gênica têm um grande número de atributos da classe, mas alguns poucos exemplos estão disponíveis. Portanto, esses conjuntos de dados podem ter o problema de alta dimensionalidade. Além disso, muitos desses conjuntos de dados são desequilibrados, ou seja, as classes não tem o mesmo número para cada categoria.

5.2 *Descrição dos conjuntos de dados*

Para avaliar o desempenho relativo dos algoritmos deste trabalho, foi utilizada uma coleção de conjunto de dados disponíveis a partir da expressão de genes de *microarray*, descrito em (GÜÇKIRAN et al., 2019). Os conjuntos de dados empregados aqui estão relacionados a diferentes tipos ou subtipos de câncer. A tarefa de classificação refere-se a rotular diferentes amostras (instâncias) de acordo com seus níveis de expressão do gene (atributo). As principais características estruturais dos conjuntos de dados são resumidas na Tabela 5.1 e as características de cada tipo de tecido de câncer são descritos na sequência.

- Câncer de Pulmão este conjunto de dados contêm 31 amostras de mesotelioma pleural maligno (MPM) e 150 amostras de adenocarcinoma (ADCA), com a expressão de 12533 genes para cada amostra. (Liu et al., 2020)
- Câncer de Próstata incluem 52 tumores de próstata (PR) e 50 amostras espécimes de próstata normais (N), com a expressão de 12600 genes para cada amostra. (Liu et al., 2020)
- 11_Tumores é conhecida na literatura como 11 tumores e inclui diferente tipos de tecidos como, 27 amostras de câncer de ovário, 26 de mama, 26 de próstata, 23 colo retal, 14 Pulmão Adeno, 14 Pulmão escamoso, 12 Gastroesôfago, 11 rim, 8 bexiga/ureter, 7 fígado, 6 pâncreas, com a expressão de 12533 genes para cada amostra. (Guillen e Ebalunode, 2016)

- Leucemia inclui 6 subtipos de leucemia 79 TEL-AML1, 64 hiperdiplóide> 50 cromossomos, 43 T-linhagem ALL, 27 E2A-PBX1, 20 rearranjos MLL e 15 BCR-ABL, com a expressão de 12625 genes para cada amostra. (Yang et al., 2020)
- SRBCT inclui 8 Linfoma de Burkitt (BL), 23 Sarcoma de Ewing (EWS), 12 neuroblastoma (NB) e 20 rabdomiossarcoma (RMS).

Base de Dados	***Chip***	***Tecido***	***Classes***	***m***	***Dist. Classes***	***d***	***n***
Câncer de Pulmão (Gordon et al., 2002)	Affy	Pulmão	2	181	150, 31	145	12533
Câncer de Próstata (Singh et al., 2002)	Affy	Próstata	2	102	52, 50	58	12600
11_Tumores (Su et al., 2001)	Affy	Multi-tecido	11	174	26, 26, 23, 14, 14, 12, 11, 8, 7, 6	140	12533
Leucemia (Yeoh et al., 2002)	Affy	Medula óssea	6	248	79, 64, 43, 27, 20, 15	210	12625
SRBCT (Khan et al., 2001)	cDNA	Multi-tecido	4	63	23, 20, 12, 8	47	2308

Tabela 5.1: Resumo dos conjuntos de dados e suas características, como tipo de chip microarray (Chip), (Tecido), número de classes (Classes), número de amostras (m), distribuição de amostras dentro das classes (Dist. Classes) e dimensionalidade após pré-processamento (d) e dimensionalidade original da base (n).

5.3 Métodos

Os algoritmos de ML e DL foram executados usando a linguagem de programação *Python* e bibliotecas *Scikit-Learn*, que são explicados em (Pedregosa et al., 2011). Estratégias de processamento de dados PCA e normalização com *Mini_Max_Scaler* foram usados com a decomposição e módulos de pré-processamento do *Scikit-Learn*. Além disso, o MLP e CNN foram implementados usando *Keras* e *Tensowflow* (Géron, 2019). E todas as imagens com os gráficos dos resultados foram criadas com a biblioteca do *matplotlib* (Tosi, 2009).

5.4 Preparação dos dados

Para os experimentos, primeiramente, os dados são embaralhados para evitar algum viés que os algoritmos possam ter e dividimos as informações em dois grupos; o primeiro grupo corresponde às características ou (vetor de atributos) e o segundo grupo às classes ou (vetor de *label*). As características compõem uma matriz de tamanho m × n e as classes são um vetor de tamanho $m \times 1$, onde m é o número de amostras e n representa as características dos genes para cada classe.

Uma das particularidades dos conjuntos de dados usados neste trabalho possui a alta dimensionalidade, como já dito antes, o número de características é maior do que o número de amostras. Portanto, os dados podem ser dispersos e os resultados não são estatisticamente estáveis ou confiáveis, afetando diretamente a precisão alcançada pelos algoritmos ML e DL.

Para resolver ou amenizar este problema, antes de alimentar os algoritmos, aplicou-se aos dados duas técnicas de pré-processamento: A normalização com a técnica *Mini_Max_Scaler*

e análise de componentes principais PCA. A primeira técnica busca para que os dados estejam em uma faixa de valores adequados para calibrar o modelo. Com a segunda técnica, reduz a dimensionalidade dos atributos e os ruídos das características irrelevantes durante o treinamento do modelo podem ser diminuídos. Os algoritmos de ML e DL podem aprender as características de um determinado problema de uma certa quantidade de dados. Esses dados são geralmente subdivididos aleatoriamente em dois grupos: treinamento e teste. Um conjunto de dados de treinamento é usado para calibrar os parâmetros do modelo, e um conjunto de dados de teste é utilizado para avaliar o desempenho do modelo (Eraslan et al., 2019).

Deste modo, a divisão inicial dos dados foi feita com o método *Holdout*. Ou seja, é subdividido aleatoriamente em dois subconjuntos (80% de treinamento e 20% de teste). Na Figura 5.2 é evidenciada a forma de como foram feitas as divisões de cada partição dos dados. E cada partição com diferentes pré-processamento de dados, sendo a primeira partição com os dados originais da base *microarray*, a segunda partição com a aplicação da normalização *Mini_Max_Scaler*, a terceira partição com a aplicação do PCA com aplicação de variância de 96% para reduzir a dimensionalidade dos dados e a última partição com ambos pré-processamento, normalização *Mini_Max_Scaler* e PCA.

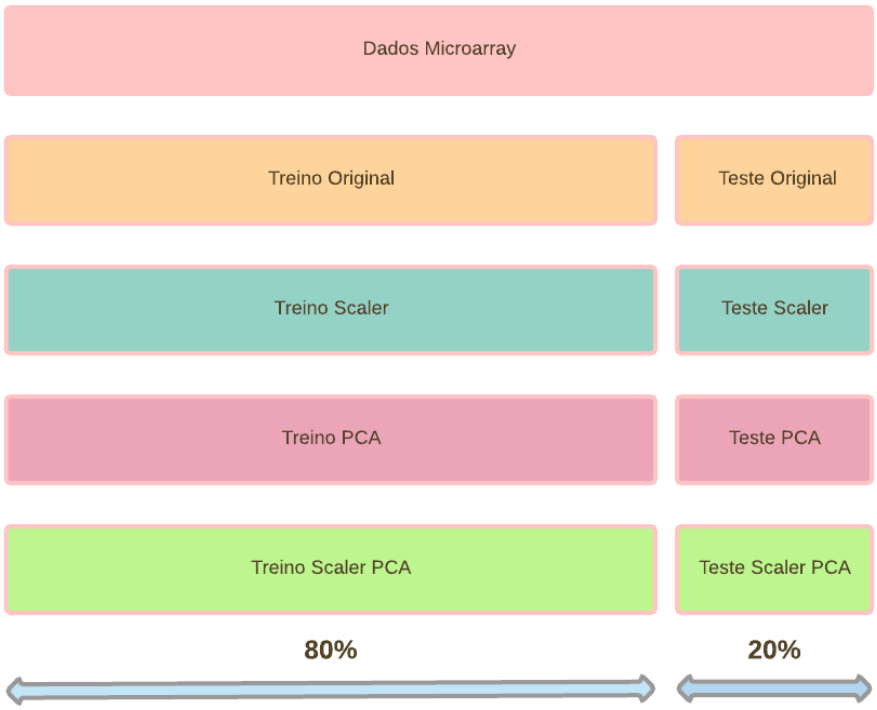


Figura 5.2: Divisão dos dados em quatro partições, cada uma com o método *Holdout* Fonte: Elaborado pelo autor.

5.5 Processo de Execução e Ajuste dos algoritmos

A execução dos algoritmos de ML foi feita em duas etapas:

- A primeira foi a calibração inicial usando o conjunto de treinamento. Para encontrar o melhor comportamento usando todos os conjuntos de dados, os experimentos consistiram em executar combinações de paramentos de cada algoritmo de ML e foram ajustados entre um determinado

intervalo de valores. Ou seja, para cada algoritmo de ML, foram executados os experimentos fazendo iterações pelos seus respectivos parâmetros e foi estabelecido um limite para cada parâmetro. Podemos conferir isto na tabela 5.2 na coluna *Range* esta descrição; Para o KNN a iteração foi para o número de amostras (K vizinhos), para os algoritmos SVM e RL a iteração é através do valor de C em 10 passos e o RF percorre a profundidade da árvore. Feito isto, são avaliadas para cada algoritmo as quatro partições de teste do pré-processamento. A Tabela 5.2 demostra os parâmetros usados para esta primeira etapa.

Algoritmos	***Parâmetros***	***Range***	***Step***	***Descrição***
KNN	*n_neighbors*	*1–m*	*1*	*Número de vizinhos*
SVM	*C,* *gamma*	*10–50* *scale*	*10*	*Parâmetro de penalidade C do termo de erro* *Gama é o parâmetro livre da função de base radial gaussiana*
LR	*C*	*0.1–1*	*0.1*	*Inverso da força de regularização*
RF	*n_estimators,* *max_depth,* *min_samples_split,* *max_features*	*40*	*1*	*Número de estimadores* *Profundidade da floresta* *Número de divisão das árvores* *Número máximo de atributos a serem considerados na divisão*

Tabela 5.2: Parâmetros dos algoritmo de ML testados, onde m é o número de amostras

- Na segunda etapa dos algoritmos de ML é realizada validação cruzada aninhada estratificada que é uma abordagem de otimização de hiperparâmetros do modelo e ao mesmo tempo, seleciona o modelo que tenta superar o problema de *overfitting* do conjunto de dados de treinamento. Esta etapa tem mais detalhes na sequência na secção *Validação cruzada*.

5.6 Implementação da Rede Neural

Dois tipos de redes foram usados para DL; a primeira é uma rede neural totalmente conectada e a segunda é uma rede neural convolucional. Dependendo da busca em grade *GridSearchCV* da biblioteca do *scikit-learn*, para cada conjunto de dados alguns parâmetros podem mudar. Um exemplo de configuração utilizada pode ser descrito como: Para o algoritmo MLP que é um *feedforward* ANN consiste em três camadas totalmente conectadas, a maior parte das bases convencionou-se através de busca por hiperparâmetros com o recurso de pesquisa em grade *GridSearchCV* por usar 50 neurônios cada, após a segunda camada densa é inserido o *Drop_out* e a função de ativação *Relu* após as camadas densas; em seguida, uma camada densa final de (*n* classes) neurônios é gerada com a função de ativação *Softmax* para gerar a probabilidade do tipo de câncer.

Para a CNN a implementação consiste em três camadas convolucionais com filtro e *kernel* apropriado para cada base de dados. O *Drop_out* é integrado depois da primeira camada com fator de abandono otimizado para cada base de dados, em seguida, a função de ativação *linea*r ou *Relu*.Um *MaxPooling* é inserido após a segunda convolução e *Flaten* após a terceira convolução. Em seguida uma camada densa com uma função de ativação *linea*r ou *Relu*, e por fim, uma camada densa de (*n* classes) neurônios com a função de ativação *Softmax* para gerar a probabilidade do tipo de câncer.

A Figura 5.3 mostra as possíveis configurações usadas para o experimento, em que o esquema à esquerda é uma MLP e o esquema à direita é uma CNN.

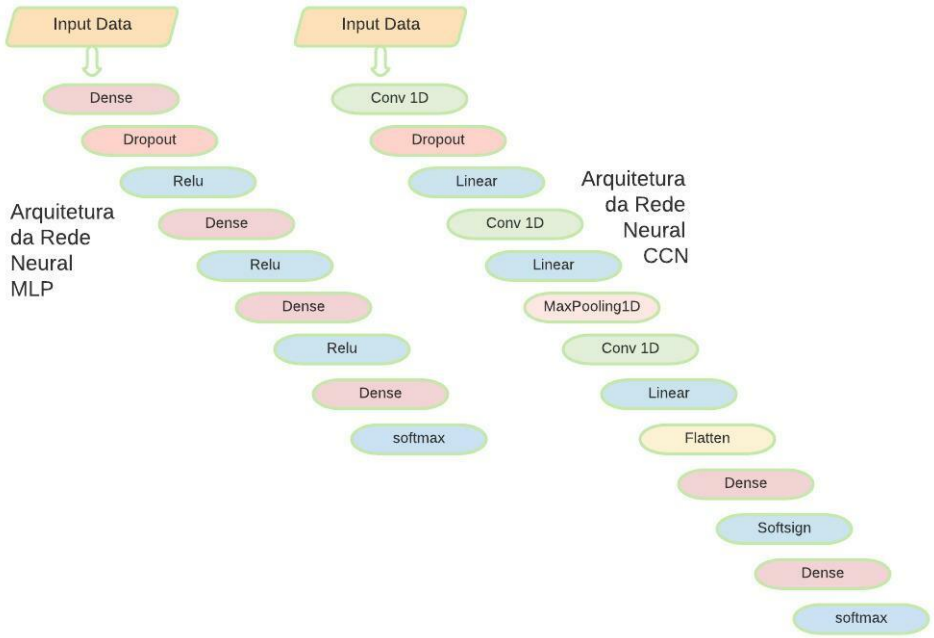


Figura 5.3: Arquiteturas da Rede Neural Artificial usadas para classificação de câncer.
Fonte: Elaborado pelo autor e inspirada em (Tabares-Soto et al., 2020) e (Shanmugamani, 2018).

Para os algoritmos de DL foi realizada a otimização para ajustar o modelo através de hiperparâmetros com o recurso de pesquisa em grade *GridSearchCV*. A razão é que as redes neurais são notoriamente difíceis de configurar e há muitos parâmetros que precisam ser definidos. Além disso, os modelos individuais podem ser muito lentos para treinar. Este método busca pelos melhores parâmetros comuns da rede neural, como taxa de aprendizagem, taxa de *Drop_out*, épocas, número de neurônios e etc. A Tabela 6.50 lista os parâmetros possíveis para a otimização da rede neural.

Parâmetros	***Valores***	***Descrição***
Batch size	*10, 20, 30, 40, 50, 60, 70, 80, 90, 100*	*Número de exemplos de treinamento utilizados em uma iteração*
Epochs	*10, 50, 100, 200*	*Número de vezes que o algoritmo de aprendizagem funcionará durante todo o treinamento*
Training optimization algorithm	*SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam*	*Ferramentas que atualizam os parâmetros do modelo e minimizam o valor da função de perda, conforme avaliado no conjunto de treinamento*
Learning rate	*0,001, 0,01, 0,1, 0,2, 0,3*	*Hiperparâmetro que controla o quanto os pesos estão sendo ajustados em relação ao gradiente de perda*
Momentum	*0,0, 0,2, 0,4, 0,6, 0,8, 0,9*	*Valor entre 0 e 1 que aumenta o tamanho dos passos dados em direção ao mínimo, tentando pular de um mínimo local*
Neuron activation function	*uniforme, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, he_uniform*	*Inicialização de pesos em camadas ocultas da rede*
Neuron activation function	*softmax, softplus, softsign, relu, tanh, sigmoid, hard_sigmoid, linear*	*Como a saída do neurônio é ativada com base em suas entradas*
Dropout regularization	*0,0, 0,1, 0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8, 0,9*	*Processo de eliminação aleatória de nós durante o treinamento*
Weight constraint	*1, 2, 3, 4, 5*	*Valor que introduz uma penalidade para a função de perda ao treinar uma rede neural para encorajar a rede a usar pequenos pesos*
Number of neurons in the hidden layers	*1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100*	*Quantidade de neurônios que compõem cada camada oculta da rede*
Kernel Size	*1,3,5,7*	*kernel se refere à largura x altura da máscara do filtro*
Filters	*1, 16, 32, 64, 128, 256*	*Quantidade de Kernel*

Tabela 5.3: Parâmetros dos algoritmos das Redes Neurais

5.7 Validação cruzada

Para medir a real precisão dos algoritmos e avaliar capacidade de generalização do modelo, foi aplicada a validação cruzada com diferentes abordagem para os algoritmos de ML e Dl.

Os algoritmos KNN, SVM, LG, RF foram treinados e validados com a mesma fração de dados e cada experimento foi repetido 5 vezes do *loop* externo e 2 do *loop* interno com a

abordagem Nested Cross-validation para buscar os melhores parâmetros e função da validação cruzada estratificada (*StratifiedKFold*) do *scikit-learn*.

Para os algoritmos MLP e CNN, foi usada a função da validação cruzada estratificada (*StratifiedKFold*) do *scikit-learn* que estima o erro de forma eficiente e imparcial, ideal para dados de *microarray* e o uso de cinco *Kfold*. Neste caso específico tanto para o DL e ML, um número maior de *Folds* não seria adequado para os dados estudados, pois alguns dos conjuntos de dados contém classes com poucas amostras e seriam sub-representados em algumas das partições. Um exemplo disto é a base de dados 11_Tumores(Su et al., 2001) onde em sua distribuição por classe o maior limite para a divisão dos *Folds* é 6, e nesta base existem 6 exemplos de uma classe.

Somente para os algoritmos de ML o ajuste do hiper-parâmetro foi realizado com o método *Nested cross-validation estratificada*. Assim dizendo, CV aninhada com 5x2, ou seja no *loop* externo a divisão é em 5 *Kfold* e *loop* interno 2 *Kfold*. Por fim, são demostrados os experimentos com Redes Neurais Profundas (DNN), como as redes neurais totalmente conectadas, MLP e CNN.

6 Resultados e Discussão

Este capítulo descreve uma discussão detalhada sobre o método proposto com base nos resultados obtidos. Primeiramente, os resultados são organizados por base de dados, ou seja, são apresentados todos os resultados dos algoritmos para cada base de dados.

São plotados gráficos de verificação da performance dos modelos onde compara as quatro bases de pré-processamento geradas, cada uma com seu treino e teste. Para tornar a visão do modelo mais intuitivo a matriz de confusão e a curvas ROC são plotadas.

Para calcular a magnitude do erro durante o treinamento e o teste, são plotados gráficos de Accuracy (Precisão) e Loss (perda) para os algoritmos de MLP e CNN. E para a visualização da comparação dos algoritmos de ML com a abordagem Nested Cross-validation estratificada e a comparação com todos os algoritmos deste experimento com a validação cruzada estratificada são plotados gráficos Box (Diagrama de caixa).

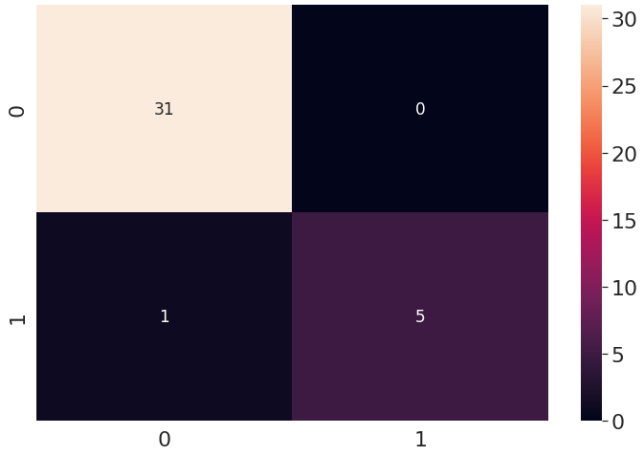
A comparação de todos os algoritmos e os gráficos demostrados, a validação cruzada estratificada e Nested Cross-validation são da base de teste Scaler PCA, pois é a partição dos dados com os dois pré-processamentos definidos para este estudo. Contudo, os resultados para as outra partições serão descritas. Os resultados dos algoritmos de ML e DL com a matriz de confusão e curva ROC são do método *holdout* com o conjunto de dados de teste que corresponde a 20% de todos os dados. As matrizes de confusão são referentes aos melhores parâmetros dos algoritmo e as que obtiveram várias confusões dos exemplos não classificados corretamente terão uma tabela mais detalhada para observar a porcentagem do acerto de cada classe.

A curva ROC é usada por classificadores binários por ser uma boa ferramenta para ver a taxa de verdadeiros positivos versus falsos positivos. Para estender a curva ROC à classificação multiclasse ou multi-rótulo, é necessário binarizar a saída e a estratégia usada é denomina um-contra-todos, ou seja, para cada classe, consideramos como a classe positiva e agrupamos as demais classes juntamente como a classe negativa.

6.1 Algoritmos com a base de dados Câncer de Pulmão

Esta base contém 181 amostras e os algoritmos usam 20% para o teste e assim restando 37 amostras para avaliar o desempenho da classificação.

• *Resultado do Algoritmo KNN*



Accuracy	*0.97*
F1-Score	*0.97*
Recall	*0.97*
Precision	*0.97*

Tabela 6.1: Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão

Figura 6.1: Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA

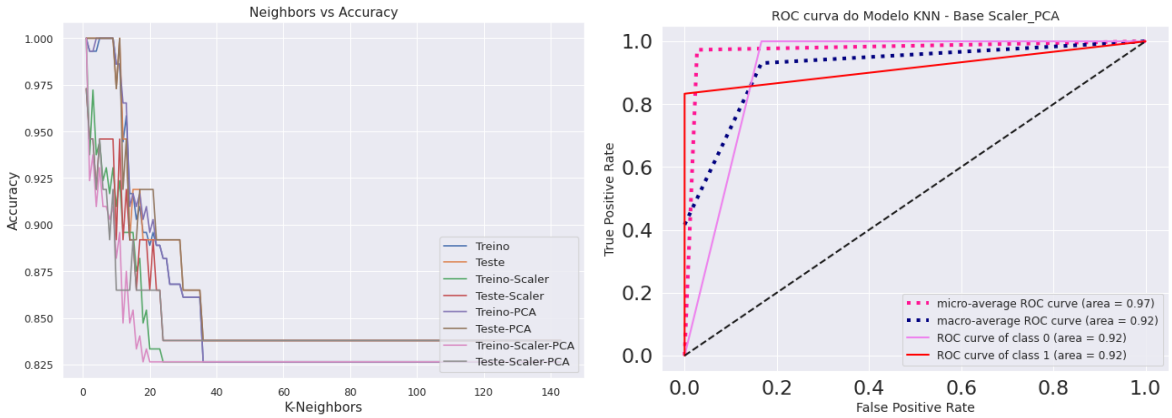


Figura 6.2: À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes

Para o algoritmo KNN percebe-se um melhor desempenho quando o K vizinho mais próximo é igual a 1, ou o menor K possível.

Esta é a razão do KNN demostrar um resultado muito bom. Mas assim que aumenta o número de vizinhos, sua *accuracy* cai bruscamente como é visto na Figura 6.2 à esquerda e à direita a curva ROC compatível com a matriz de confusão que mostra amostras de teste classificadas de forma correta e ou incorreta. A base de dados Câncer de Pulmão possui 150 amostras originais e 37 são usadas para o teste e se caracteriza por possuir um desequilíbrio de classes. A classe 0 possui 31 amostras e a classe 1, 6 amostras e suas pontuações são de 98% e 91% respectivamente pela métrica *f1-score* que é sensível a classes desbalanceadas. À direita, a curva ROC mostra o desempenho para cada classe, no eixo x a taxa de falso positivo FPR e no eixo y a taxa de verdadeiro positivo TPR são representadas.

• *Resultado do Algoritmo SVM*

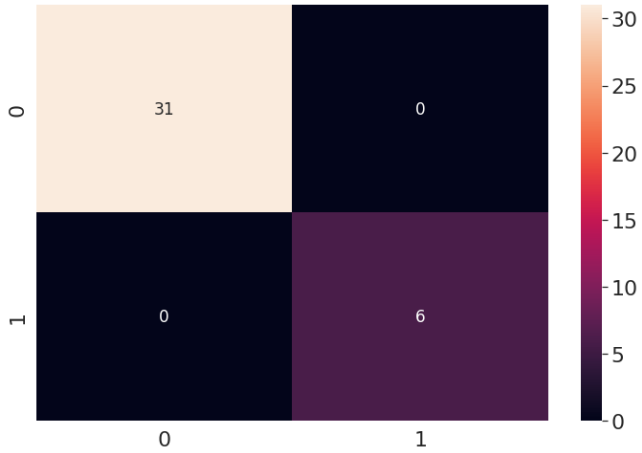


Figura 6.3: Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*1.00*
F1-Score	*1.00*
Recall	*1.00*
Precision	*1.00*

Tabela 6.2: Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão

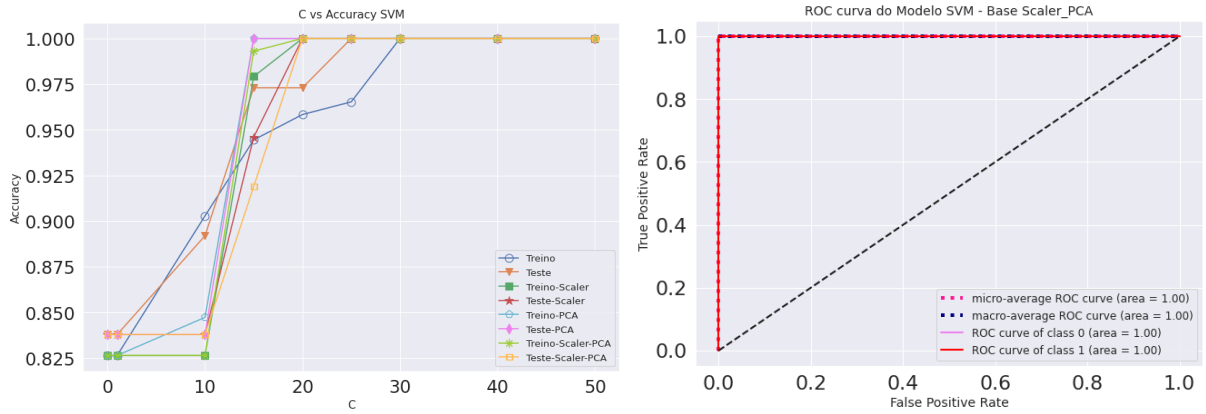


Figura 6.4: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

O SVM se comportou muito bem tanto para matriz de confusão quanto para a curva ROC atingindo 100% de precisão. Nota-se na Figura 6.4 que a *accuraccy* vai a 100% quando o valor de C atinge em média o valor de 20 na escala do gráfico. À direita, a curva ROC mostra o desempenho para cada classe, no eixo x a taxa de falso positivo FPR e no eixo y a taxa de verdadeiro positivo TPR são representadas e alcançou uma precisão de classificação perfeita neste conjunto de dados.

- *Resultado do Algoritmo RL*

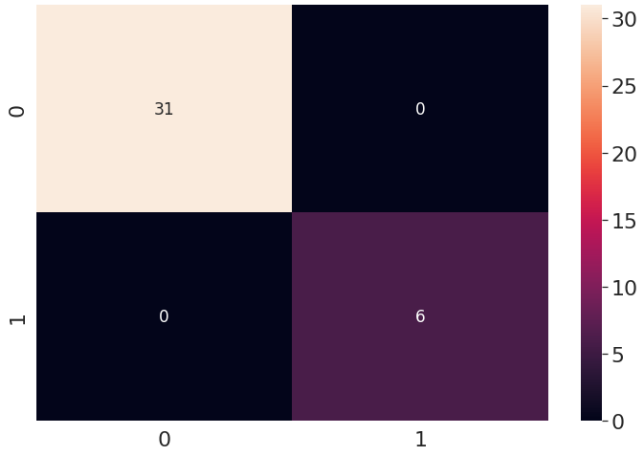


Figura 6.5: Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*1.00*
F1-Score	*1.00*
Recall	*1.00*
Precision	*1.00*

Tabela 6.3: Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão

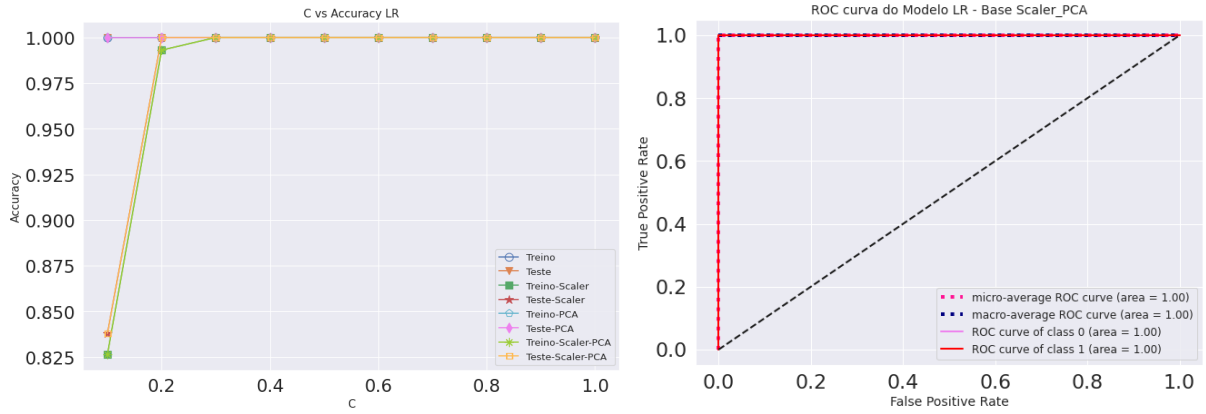


Figura 6.6: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

O algoritmo RL teve um desempenho semelhante ao SVM atingindo 100% de accuracy. Estes resultados são aceitáveis e compatível com a literatura visto em (Chou e Ko, 2017) e (GÜÇKIRAN et al., 2019) com 99% e 100% respectivamente.

- *Resultado do Algoritmo RF*

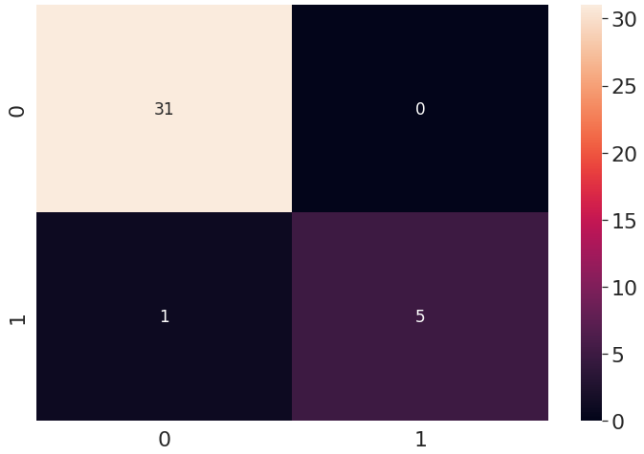


Figura 6.7: Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA

Classes	***0***	***1***
Precision	*97%*	*100%*
Recall	*100%*	*83%*
F1-Score	*98%*	*91%*
Accuracy	*97%*	

Tabela 6.4: Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão

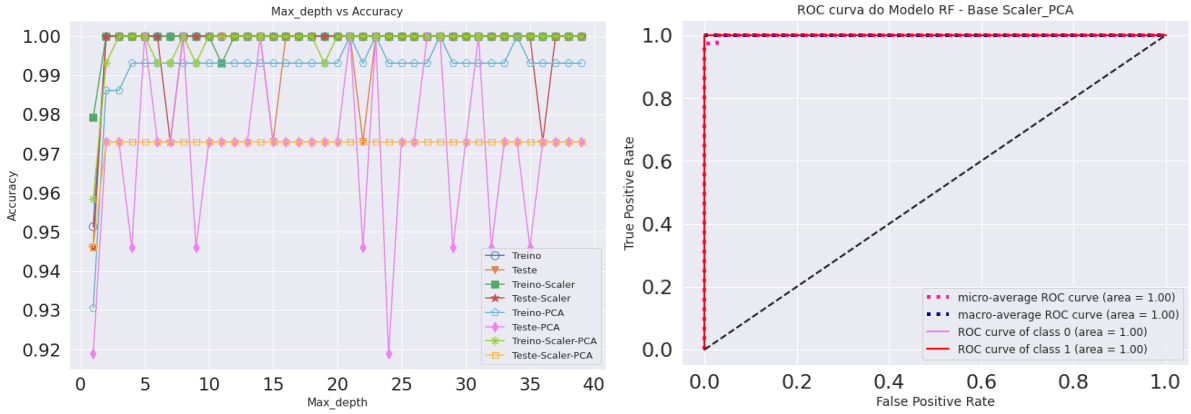


Figura 6.8: À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes

O algoritmo RF para a matriz de confusão com a classe 0 atinge 98% e a classe 1, 91% e nota-se uma instabilidade com o teste PCA demostrando um desempenho um pouco abaixo do teste Scaler PCA em relação aos testes que não possui o pré-processamento PCA.

• *Comparação dos Algoritmos de ML com a abordagem Nested Cross-validation estratificada*

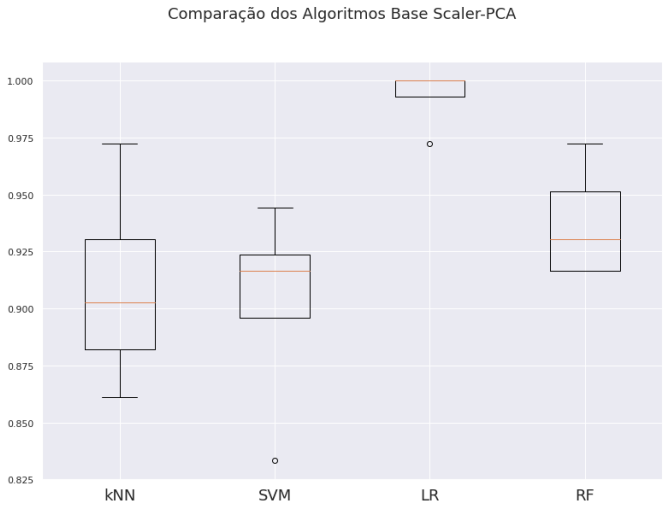


Figura 6.9: Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA

KNN	*Neighbors:3*	*90%*
SVM	*C: 10.0* *kernel: linear*	*88%*
RL	*C : 0.5* *penalty: l1* *max_iter: 500*	*99%*
RF	*n_jobs: -1* *n_estimators: 800* *min_samples_split: 10* *min_samples_leaf: 2* *max_features: sqrt* *max_depth: 50* *criterion: gini* *bootstrap: False*	*88%*

Tabela 6.5: Scores dos algoritmos e seus respectivos parâmetros na base de dados Câncer de Pulmão

No método *holdout* os valores do SVM e RL ficaram com uma pontuação perfeita, KNN e RL ambos com 97%. Uma hipótese é que pelo método *holdout* o algoritmo pode tender a ser muito otimista ou estar enviesado.

A validação cruzada em geral, estima o real desempenho do modelo, pois se comporta de maneira imparcial e a abordagem estratificada permite que pelo menos um exemplo de uma classe esteja representada em uma partição. E a CV para a base Câncer de Pulmão obteve um desempenho abaixo do método *holdout*, mas considerado muito bom.

• *Resultado do Algoritmo MLP*

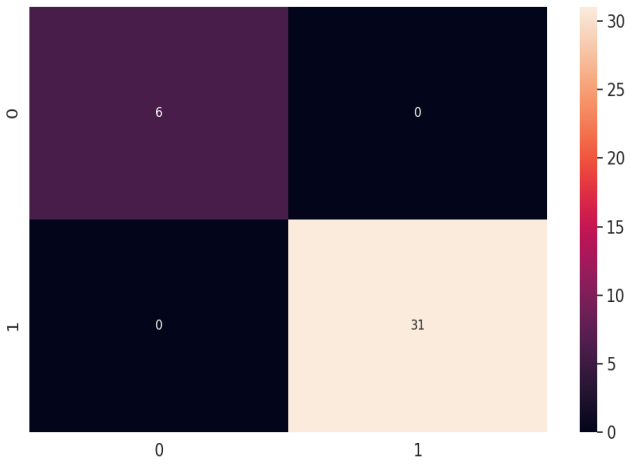


Figura 6.10: Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*1.00*
F1-Score	*1.00*
Recall	*1.00*
Precision	*1.00*

Tabela 6.6: Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão

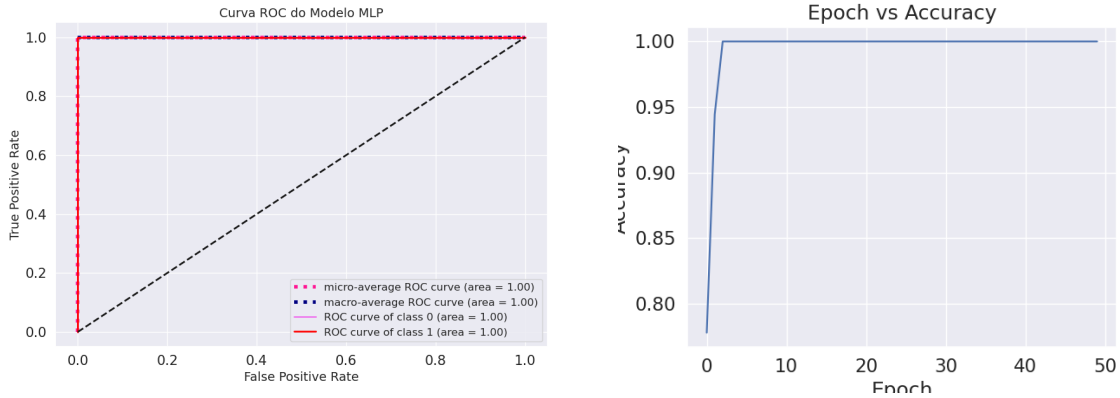


Figura 6.11: À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta.

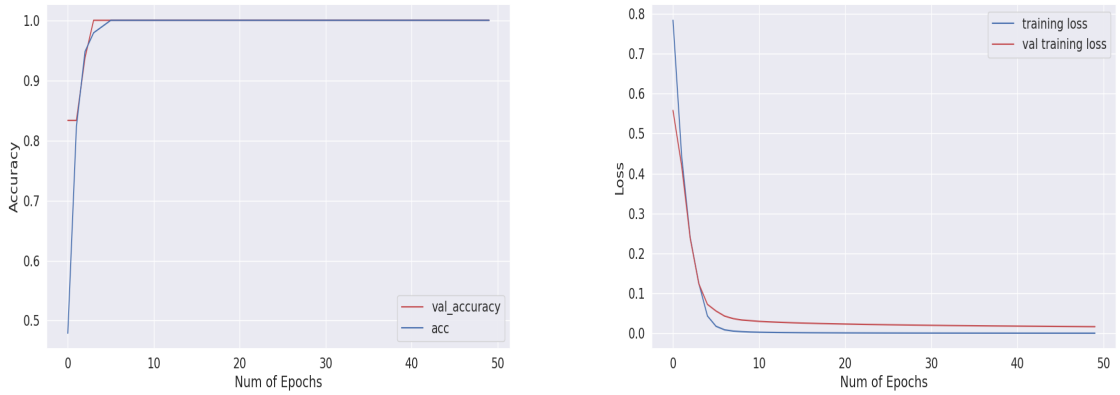
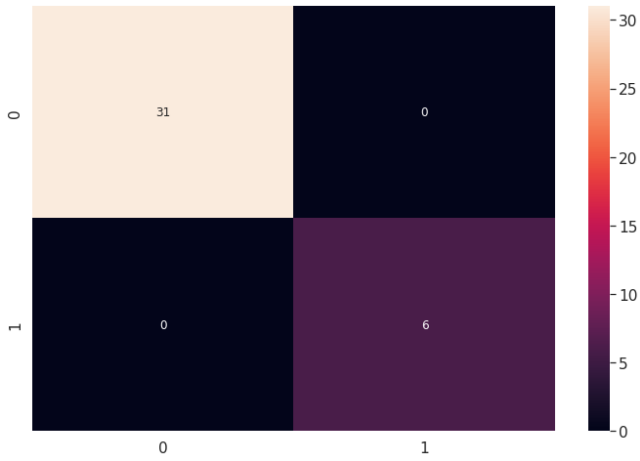


Figura 6.12: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

• *Resultado do Algoritmo CNN*



Accuracy	*1.00*
F1-Score	*1.00*
Recall	*1.00*
Precision	*1.00*

Tabela 6.7: Predição com as métricas da matriz de confusão com a base de dados Câncer de Pulmão

Figura 6.13: Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA

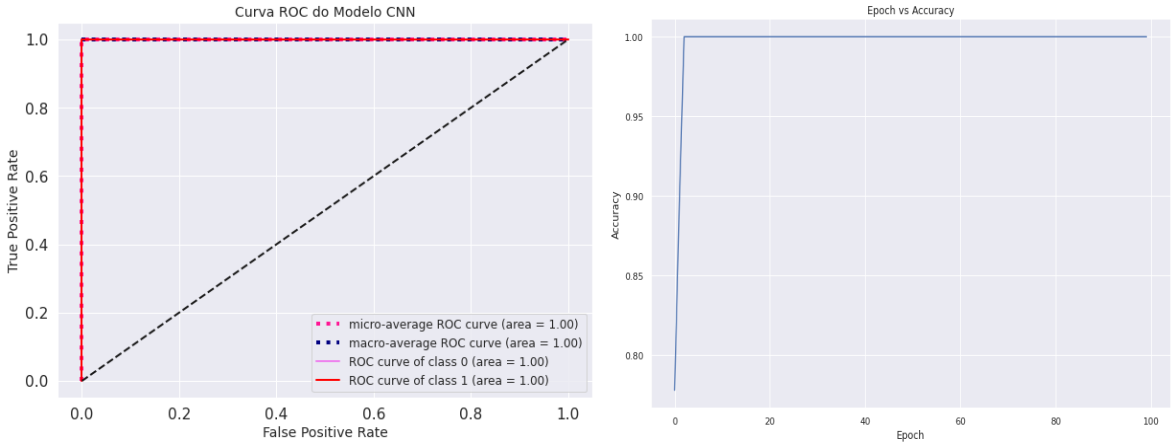


Figura 6.14: À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta.

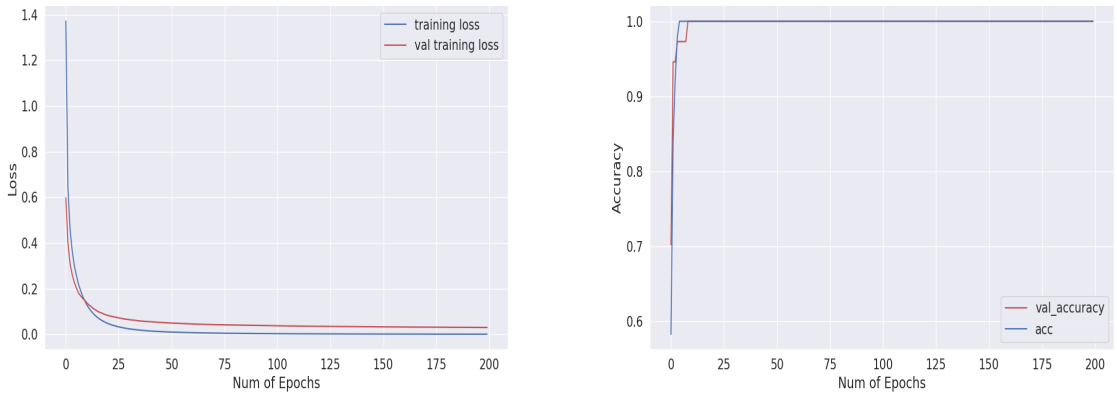


Figura 6.15: À esquerda, o comportamento da *accuracy* do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

Na Figura 6.12 e na Figura 6.15 do MLP e CNN respectivamente os gráficos de Accuracy (Precisão) e Loss (perda) que é utilizada para medir o funcionamento da rede em cada época mostram que a precisão e a perda estão com suas linhas de treinamento e validação quase juntas indicando um bom ajuste do modelo. Quando ambos os conjuntos de dados mostram resultados muito distantes, a arquitetura pode estar superdimensionada onde pode ocorrer a indicação de overffiting. O desempenho e o comportamento de convergência dos modelos sugerem que o erro quadrático médio é uma boa combinação para uma rede neural que está aprendendo esse problema.

Parâmetros	***MLP***	***CNN***
Batch size	*10*	*10*
Epochs	*50*	*100*
Training optimization algorithm	*Adam*	*SGD*
Learning rate	*0.1*	*0.01*
Momentum	*N/A*	*0,9*
Network weight initialization	*N/A*	*lecun_uniform*
Neuron activation function	*relu*	*relu*
Dropout regularization	*0,4*	*0,5*
Weight constraint	*N/A*	*4*
Number of neurons in the hidden layers	*50*	*100*
Filters	*N/A*	*64*
Kernel Size	*N/A*	*3*

Tabela 6.8: Parâmetros dos algoritmos das Redes Neurais para a base de dados Câncer de Pulmão

Os resultados para MLP e CNN com o método Holdout demostram a mesma tendência dos algoritmos SVM e RL para a base de dados Câncer de Pulmão atingindo 100% de accuracy. Estes resultados para a DL são aceitáveis e compatível com os resultados da literatura como se é visto em Basavegowda e Dagnew (2020) um estudo recente e que obteve resultados semelhante ao encontrado neste trabalho.

• *Comparação dos Algoritmos de ML e DL com a Validação Cruzada Estratificada*

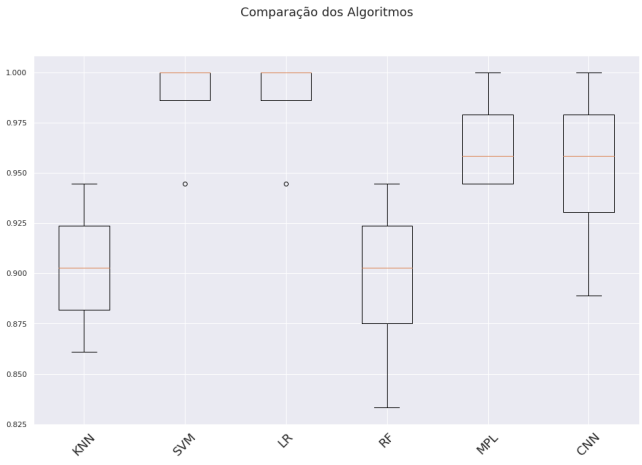


Figura 6.16: Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA

KNN	*0.92*
SVM	*0.99*
RL	*0.99*
RF	*0.91*
MLP	*0.94*
CNN	*0.95*

Tabela 6.9: Scores dos algoritmos com o método validação cruzada estratificada na base de dados Câncer de Pulmão

Para comparar a performance dos classificadores, os algoritmos foram configurados com seus melhores parâmetros e então foi realizada a validação cruzada estratifica para estimar o real desempenho de generalização dos modelos pois esta avaliação se comporta de maneira imparcial. E nota-se um desempenho pouco abaixo do método anterior Holdout, mas continua apresentando um bom desempenho. Nota-se também na Figura 6.16 a detecção de outliers (ruídos) nos modelos SVM e LG. Em estatísticas, um outlier é um ponto de observação que está distante de outras observações. Mas nem todo outlier pode ser considerado como um valor

errado. Nem a presença nem a ausência do outlier no gráfico mudariam a performance dos modelos SVM e LG. A maioria das estatísticas paramétricas, como médias, desvios padrão e correlações, e todas as estatísticas com base nelas, são altamente sensíveis a outlier. Apesar disso, não é aceitável descartar uma observação apenas porque é um outlier. Elas podem ser observações legítimas e é importante investigar a natureza do outlier antes de decidir se deve descartá-lo ou não (Aggarwal, 2017).

Modelo do Pré-processamento	***KNN***	***SVM***	***RL***	***RF***	***MLP***	***CNN***
Base Original	*100%*	*100%*	*100%*	*100%*	*100%*	*100%*
Base Scaler	*97%*	*100%*	*100%*	*97%*	*100%*	*100%*
Base PCA	*100%*	*100%*	*100%*	*100%*	*100%*	*100%*
Base PCA Scaler	*97%*	*100%*	*100%*	*97%*	*100%*	*100%*

Tabela 6.10: Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados Câncer de Pulmão

Em geral todos os algoritmos obtiveram um bom desempenho independente das diferentes partições de pré-processamento. O RF onde no teste Scaler PCA obteve uma pouca diferença de outros testes. E ressaltar que o KNN obteve um bom desempenho por conta dos primeiros vizinhos mais próximos.

6.2 Algoritmos com a base de dados Câncer de Próstata

Esta base contém 102 amostras e os algoritmos usam 20% para o teste e assim restando 21 amostras para avaliar o desempenho da classificação.

• *Resultado do Algoritmo KNN*

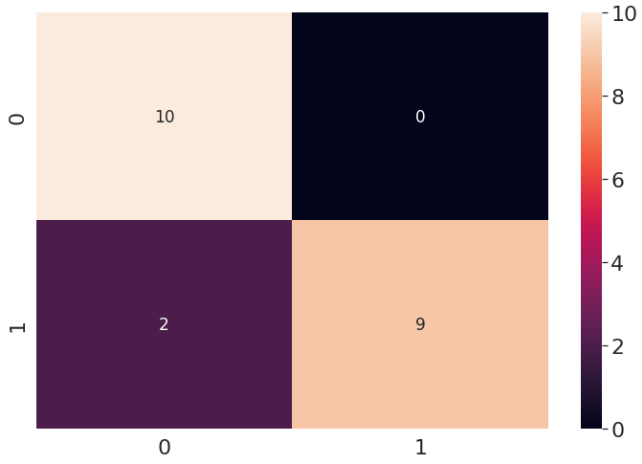


Figura 6.17: Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA

Classes	***0***	***1***
Precision	*83%*	*100%*
Recall	*100%*	*82%*
F1-Score	*91%*	*90%*
Accuracy	*90%*	

Tabela 6.11: Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata

A matriz de confusão do KNN Figura 6.17 se refere ao vizinho K=2.

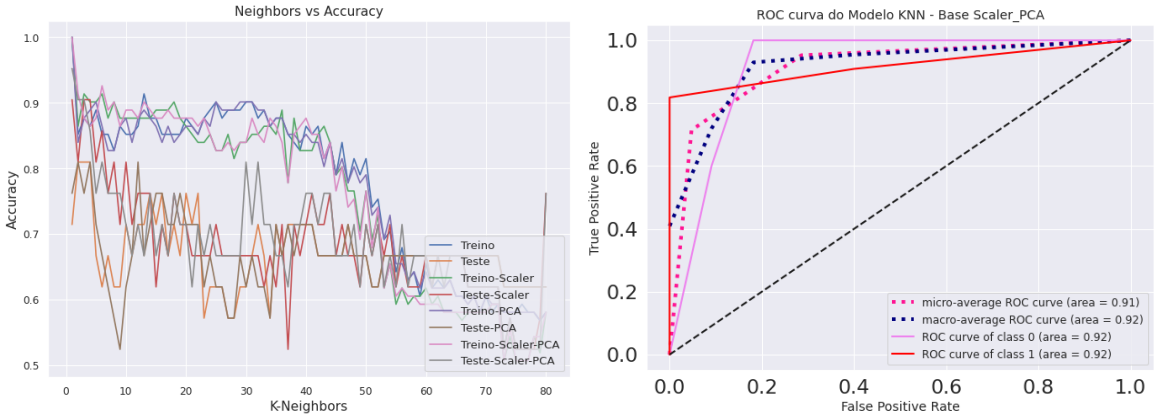


Figura 6.18: À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes

• *Resultado do Algoritmo SVM*

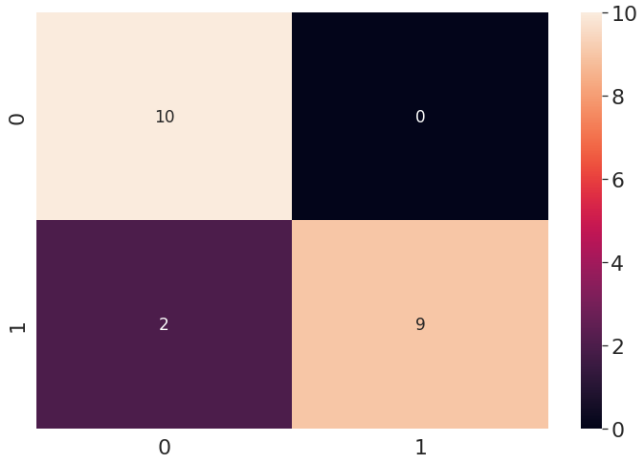


Figura 6.19: Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.90*
F1-Score	*0.90*
Recall	*0.90*
Precision	*0.92*

Tabela 6.12: Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata

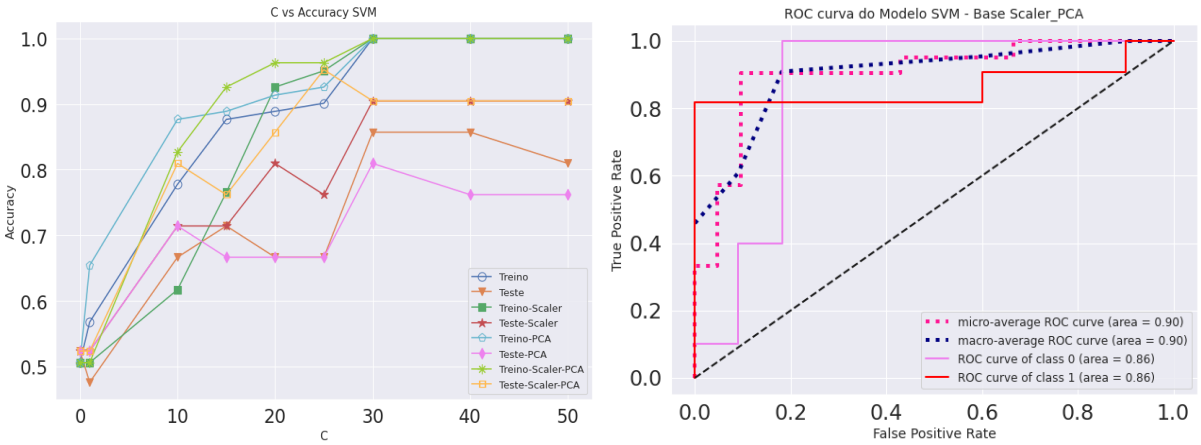


Figura 6.20: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

A matriz de confusão do SVM da Figura 6.12 se refere ao C=25 como é visto à esquerda na Figura 6.20 onde a base teste Scaler PCA atinge este valor e à direita a sua curva ROC que descreve o desempenho por classes. Como a base de dados Câncer de Próstata possui em seus exemplos classes balanceadas, o SVM atribuiu o mesmo valor para cada uma das classes.

Em todos os gráficos da curva ROC podemos notar as métricas:

- Macro - Calcula a métrica para cada classe e pega a média não ponderada.
- Micro - calcula a métrica globalmente contando o total de verdadeiros positivos, falsos negativos e falsos positivos (independente das classes).

• *Resultado do Algoritmo RL*

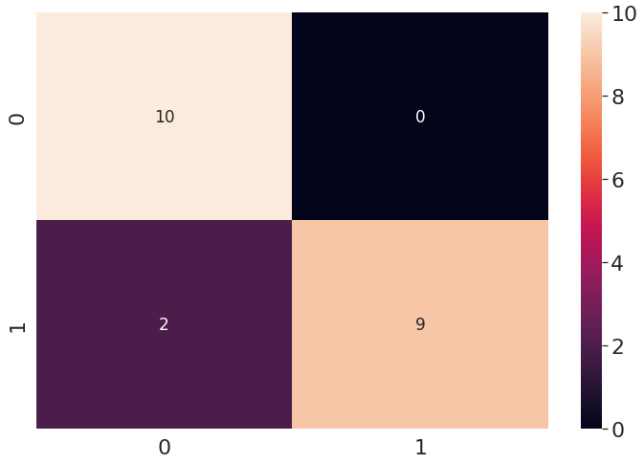


Figura 6.21: Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.90*
F1-Score	*0.90*
Recall	*0.90*
Precision	*0.92*

Tabela 6.13: Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata

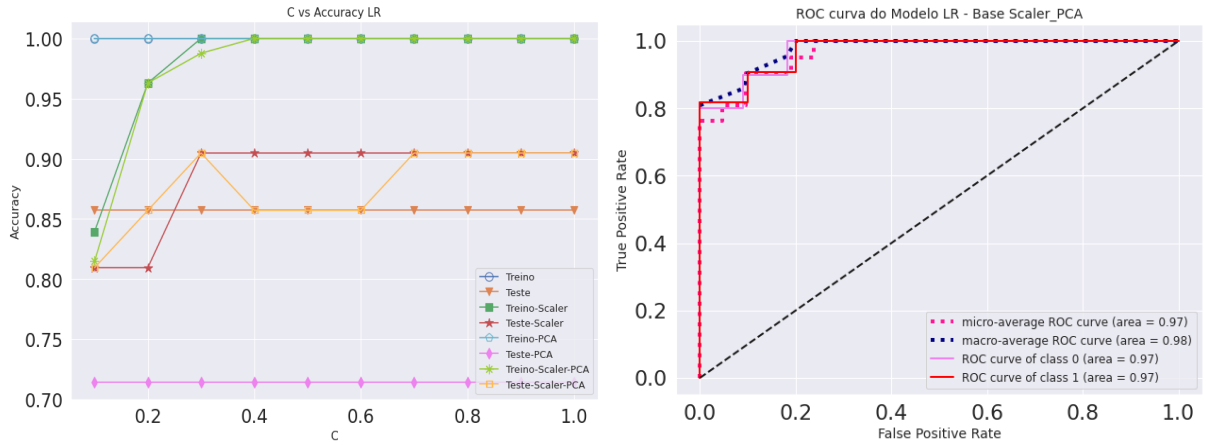


Figura 6.22: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

A base de dados de Câncer de Próstata possui os dados balanceados, porém seus dados demostram ser levemente ruidosos. Os algoritmos KNN, SVM e RL obtiveram o mesmo desempenho na matriz de confusão 91% da classe 0 e 90% da classe 1 pela métrica *f1-score*.

À esquerda na Figura 6.22 a base de teste PCA ficou bem abaixo em relação aos outros testes, à direita, a curva ROC mostra o desempenho para cada classe, no eixo x a taxa de falso positivo FPR e no eixo y a taxa de verdadeiro positivo TPR são representadas.

• *Resultado do Algoritmo RF*

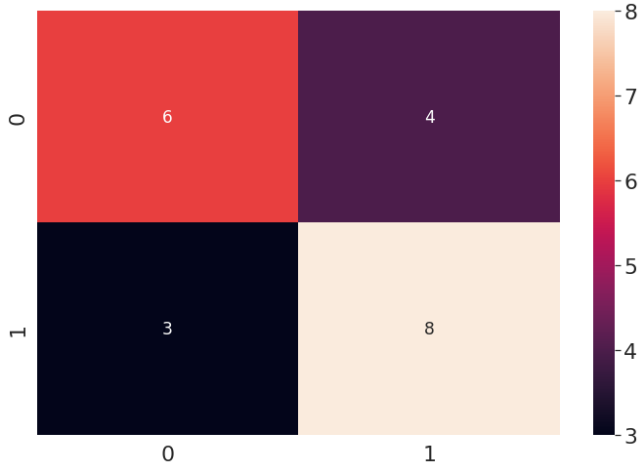


Figura 6.23: Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA

Classes	***0***	***1***
Precision	*67%*	*67%*
Recall	*60%*	*73%*
F1-Score	*63%*	*70%*
Accuracy	*67%*	

Tabela 6.14: Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata

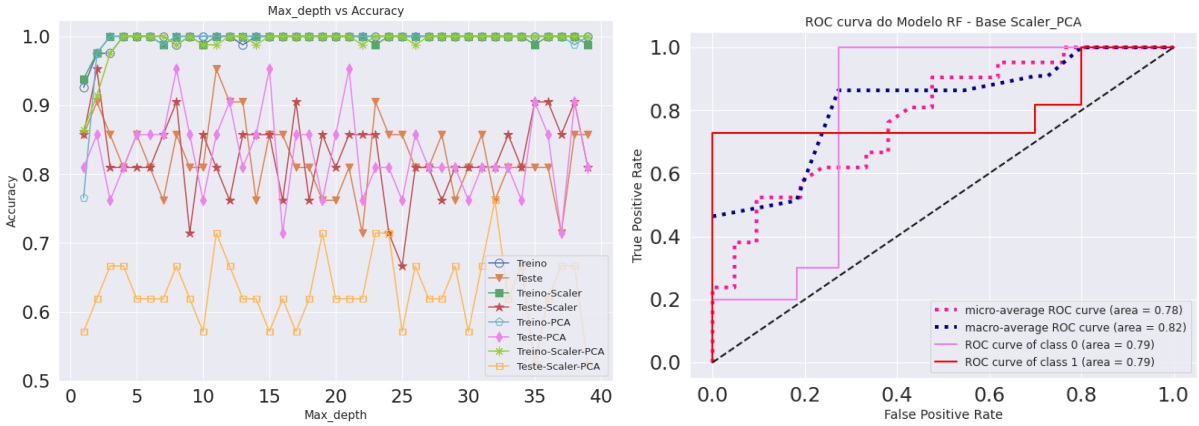


Figura 6.24: À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes

O algoritmos RF obteve o pior desempenho com a combinação da característica da base de dados Câncer de Próstata e com o pré-processamento Scaler PCA.

• *Compararação dos Algoritmos de ML com a abordagem Nested Cross-validation estratificada*

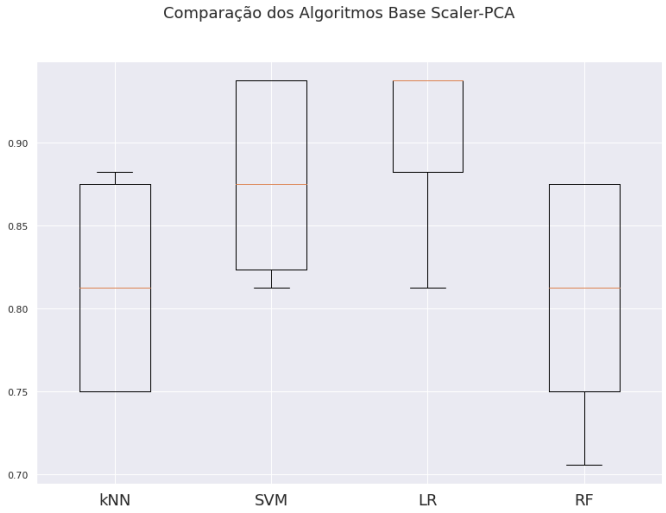


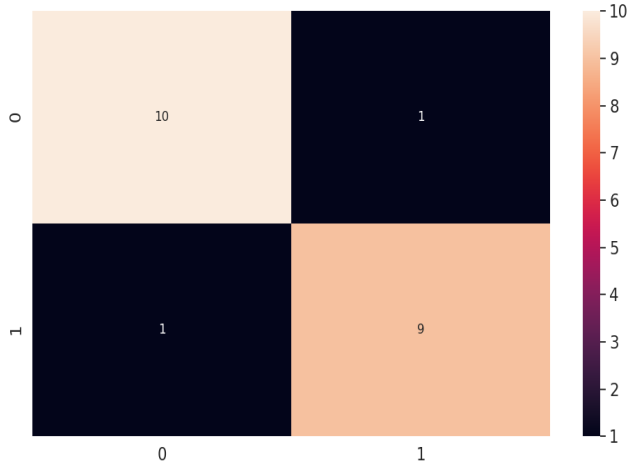
Figura 6.25: Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA

KNN	*Neighbors:14*	*81%*
SVM	*C: 0.1* *kernel: linear*	*85%*
RL	*C : 10* *penalty: l1* *max_iter: 500*	*90%*
RF	*n_jobs: -1* *n_estimators: 400* *min_samples_split: 25* *min_samples_leaf: 4* *max_features: auto* *max_depth: 20* *criterion: gini* *bootstrap: False*	*82%*

Tabela 6.15: Scores dos algoritmos e seus respectivos parâmetros na base de dados Câncer de Próstata

Com a validação cruzada Nested Cross-validation com 5 *fold* do laço externo e 2 *fold* do laço interno é comparado desempenho dos algoritmos. Este método estima a real performance e busca pelos melhores parâmetros. Neste método o algoritmo RF atingiu uma melhora, de 67% para 85% na classificação dos dados.

- *Resultado do Algoritmo MLP*



Accuracy	*0.90*
F1-Score	*0.90*
Recall	*0.91*
Precision	*0.92*

Tabela 6.16: Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata

Figura 6.26: Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA

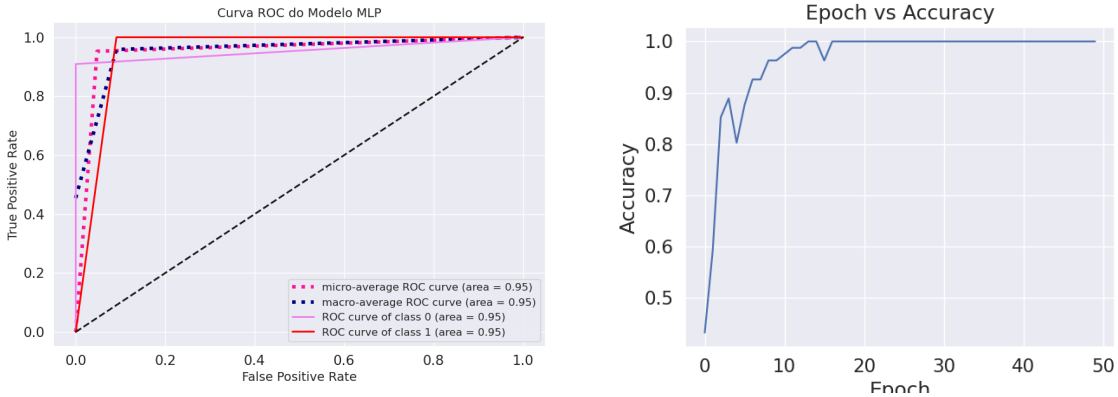


Figura 6.27: À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta.

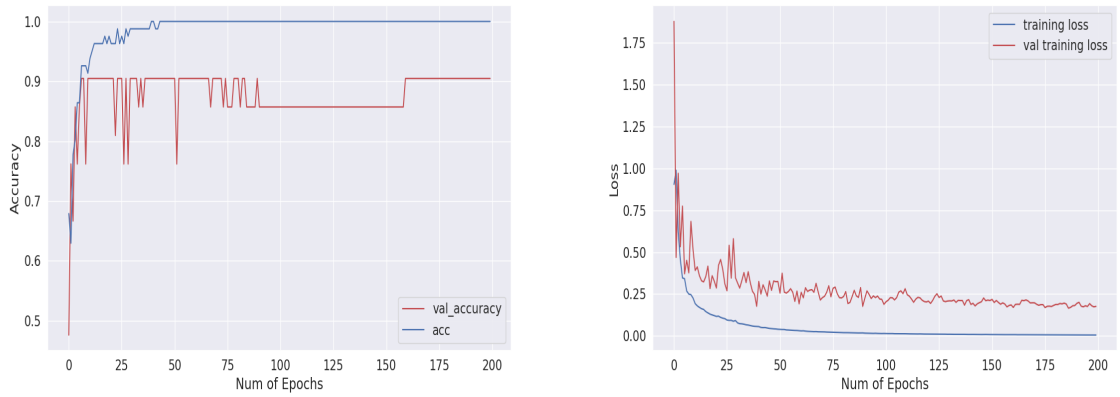


Figura 6.28: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

- *Resultado do Algoritmo CNN*

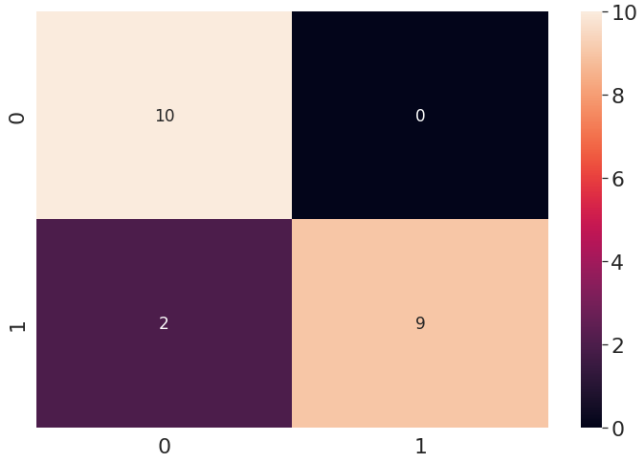


Figura 6.29: Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.90*
F1-Score	*0.90*
Recall	*0.91*
Precision	*0.92*

Tabela 6.17: Predição com as métricas da matriz de confusão com a base de dados Câncer de Próstata

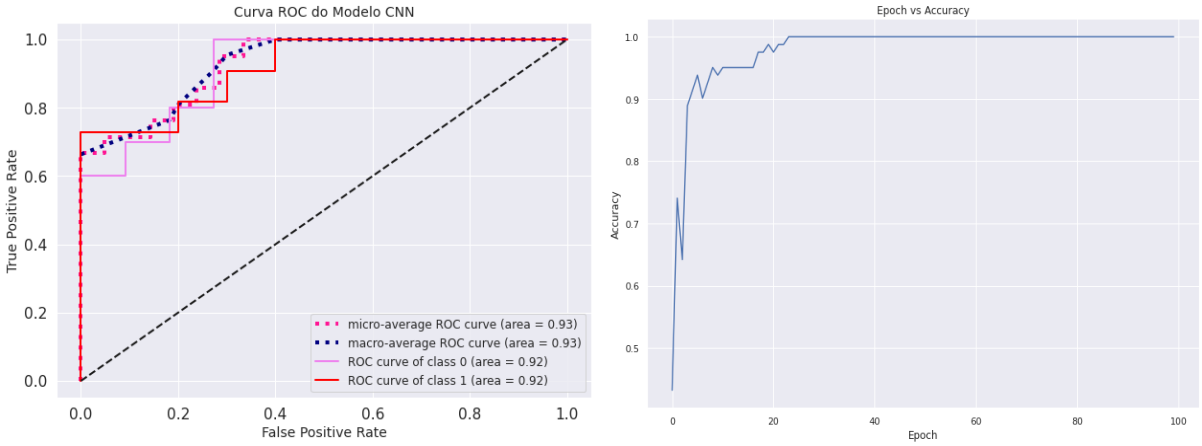


Figura 6.30: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

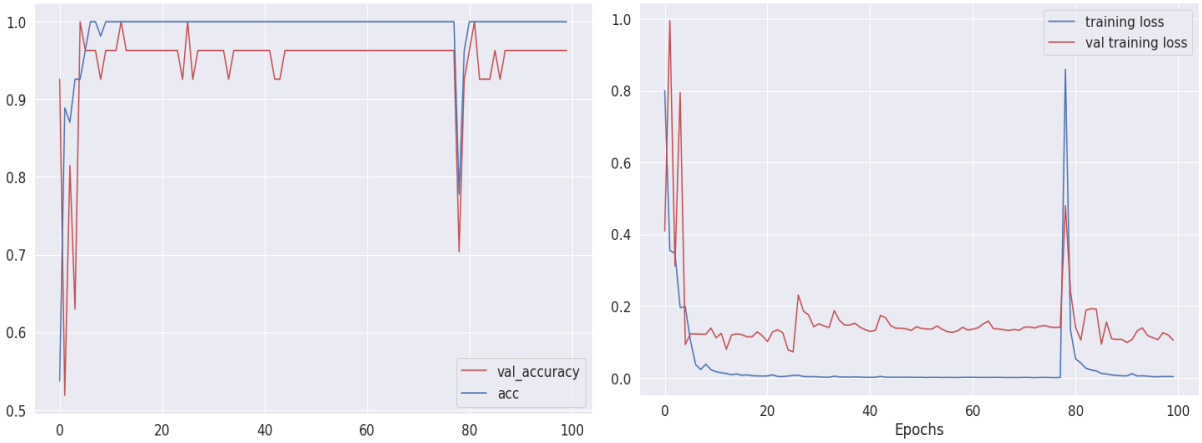


Figura 6.31: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

As Figuras 6.28 e 6.31 da MLP e CNN mostram a comparação da precisão de treinamento e validação, e também mostra as suas curvas de perda de treinamento com a perda da validação. Os gráficos mostram uma lacuna mínima entre os linhas de treinamento e casos de teste. A precisão do treinamento aumenta com o aumento do número de épocas, enquanto a perda de treinamento diminui a cada época. Isso é o que deve ser esperado quando executa a otimização de gradiente descendente. Em outras palavras, o erro deve ser reduzido a cada iteração. No entanto, chega a um momento em que a rede se estabiliza e o aprendizado se mantém constante.

Parâmetros	***MLP***	***CNN***
Batch size	*10*	*10*
Epochs	*200*	*100*
Training optimization algorithm	*Adam*	*Adam*
Learning rate	*0.1*	*0.1*
Momentum	*N/A*	*0,0*
Network weight initialization	*N/A*	*lecun_uniform*
Neuron activation function	*relu*	*relu*
Dropout regularization	*0,5*	*0,5*
Weight constraint	*N/A*	*5*
Number of neurons in the hidden layers	*50*	*100*
Filters	*N/A*	*32*
Kernel Size	*N/A*	*3*

Tabela 6.18: Parâmetros dos algoritmos das Redes Neurais para a base de dados Câncer de Próstata

• *Comparação dos Algoritmos de ML e DL com a Validação Cruzada Estratificada*

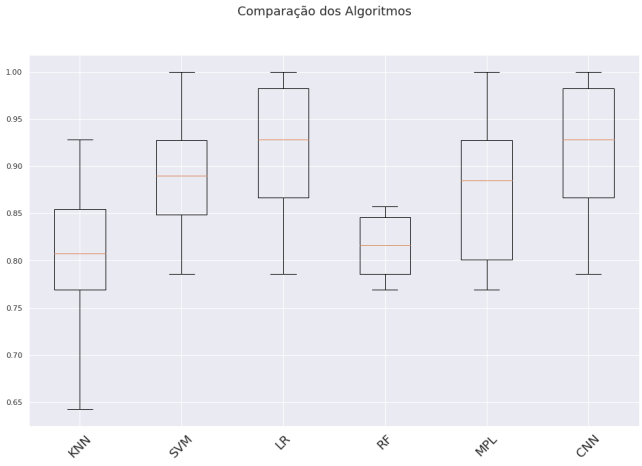


Figura 6.32: Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA

KNN	*0.80*
SVM	*0.89*
RL	*0.91*
RF	*0.81*
MLP	*0.87*
CNN	*0.91*

Tabela 6.19: Scores dos algoritmos com o método validação cruzada estratificada na base de dados Câncer de Próstata

A figura 6.32 mostra a comparação dos algoritmos para a base de dados de Câncer de Próstata e indica que os melhores algoritmos são a RL e CNN e com bom desempenho, o segundo melhor para SVM e MLP, e resultados semelhantes para KNN e RF com desempenho aceitáveis para esta base de dados. E a Tabela 6.20 descreve a accuracy dos algoritmos com o método holdout para cada partição de teste do pré-processamento.

Modelo do Pré-processamento	***KNN***	***SVM***	***RL***	***RF***	***MLP***	***CNN***
Base Original	*80%*	*80%*	*85%*	*80%*	*80%*	*87%*
Base Scaler	*80%*	*90%*	*90%*	*76%*	*85%*	*93%*
Base PCA	*80%*	*76%*	*71%*	*76%*	*87%*	*92%*
Base PCA Scaler	*90%*	*90%*	*90%*	*61%*	*90%*	*90%*

Tabela 6.20: Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados Câncer de Próstata

6.3 Algoritmos com a base de dados 11_Tumores

Este conjunto de base de dados é conhecida na literatura como 11 tumores e muito estudada pela característica se ser multi-classes com diferente tipo de câncer. Alguns dos trabalhos podemos encontrar em (Alanni et al., 2019), (Dabba et al., 2021), (Guillen e Ebalunode, 2016), (Tabares-Soto et al., 2020) e muito outros.

Esta base contém 174 amostras e os algoritmos usam 20% para o teste e assim restando 35 amostras para avaliar o desempenho da classificação.

- *Resultado do Algoritmo KNN*

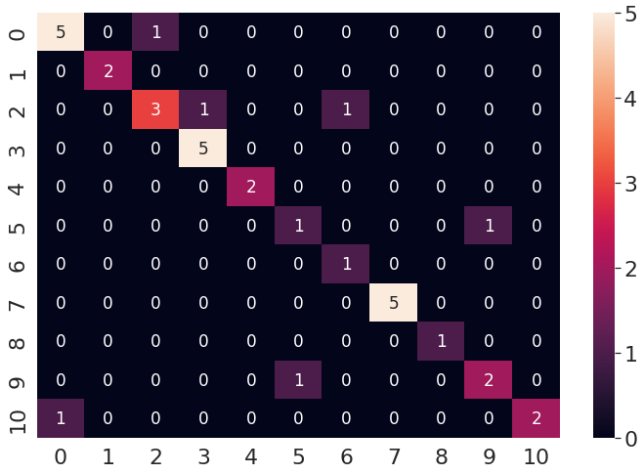


Figura 6.33: Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.82*
F1-Score	*0.82*
Recall	*0.82*
Precision	*0.84*

Tabela 6.21: Predição com as métricas da matriz de confusão com a base de dados 11_Tumores

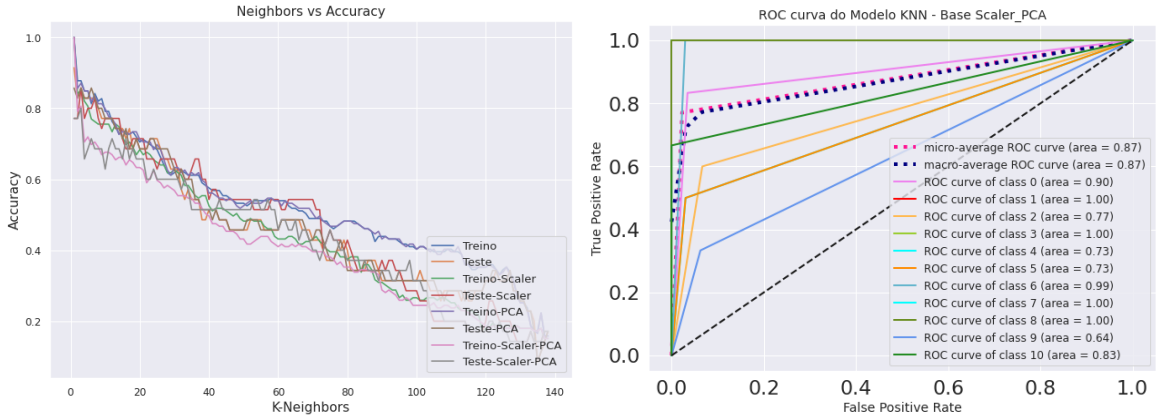


Figura 6.34: À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes

A matriz de confusão da Figura 6.33 se refere ao vizinho K=3. O algoritmo KNN obtém um bom desempenho com poucos vizinhos mais próximos, assim que aumenta seus vizinhos a sua accuracy cai como é visto à esquerda da Figura 6.34 e à direita mostra a curva ROC para cada classe usando o método um contra todos para simular um problema de classificação binária. Observe que, a classe 9 obteve o pior desempenho.

- *Resultado do Algoritmo SVM*

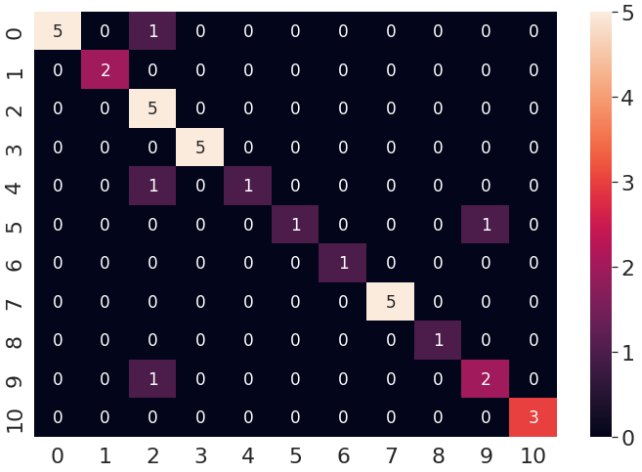


Figura 6.35: Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.88*
F1-Score	*0.88*
Recall	*0.82*
Precision	*0.89*

Tabela 6.22: Predição com as métricas da matriz de confusão com a base de dados 11_Tumores

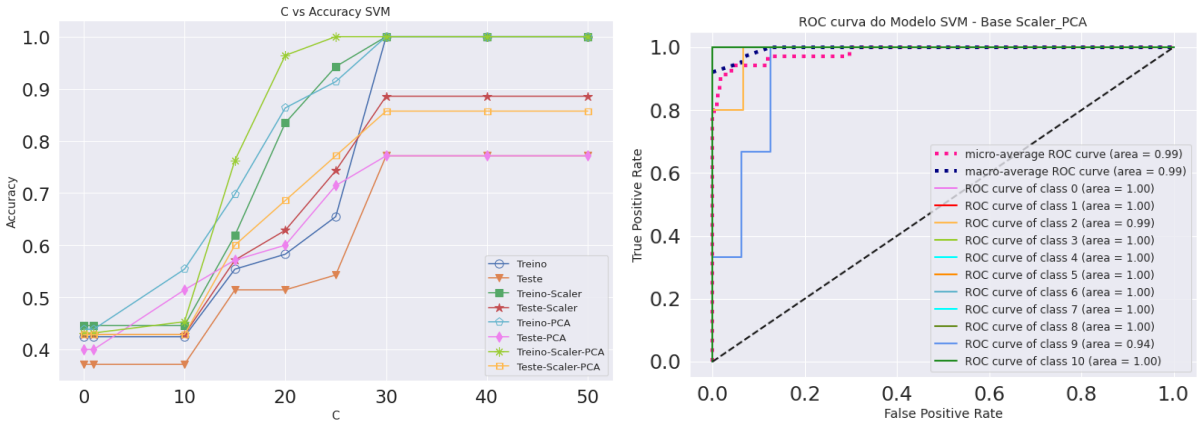


Figura 6.36: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

A matriz de confusão do SVM da Figura 6.22 se refere ao valor de C=30 como é visto na Figura 6.36 onde a base teste Scaler PCA atinge este valor.

• *Resultado do Algoritmo RL*

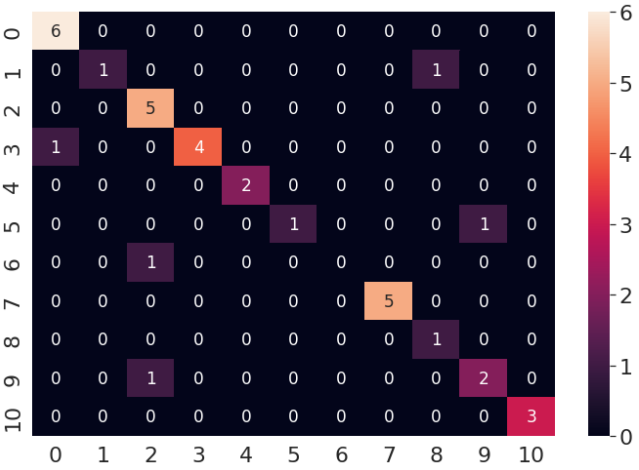


Figura 6.37: Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.85*
F1-Score	*0.84*
Recall	*0.85*
Precision	*0.86*

Tabela 6.23: Predição com as métricas da matriz de confusão com a base de dados 11_Tumores

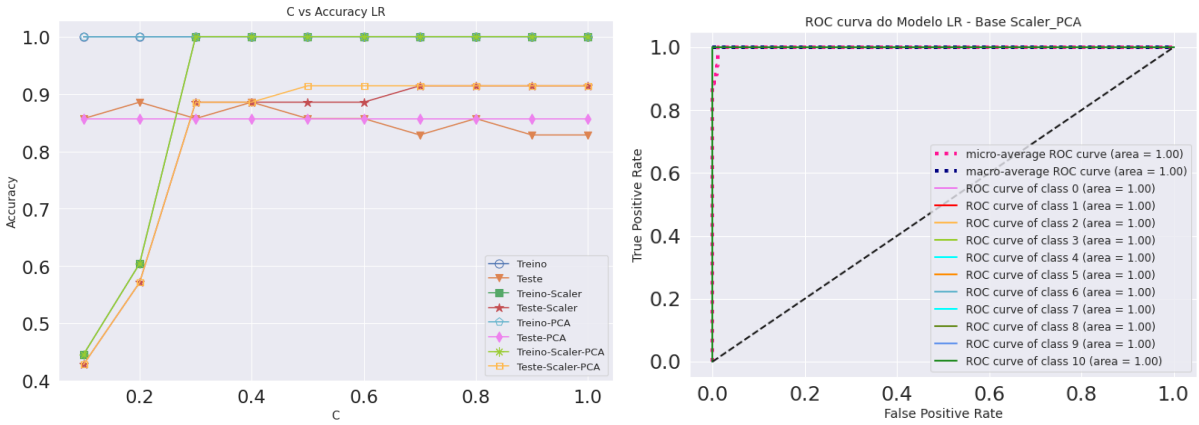


Figura 6.38: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

A matriz de confusão RL da Figura 6.37 se refere ao C=0.5 como é visto na Figura 6.38 onde a base teste Scaler PCA atinge este valor.

• *Resultado do Algoritmo RF*

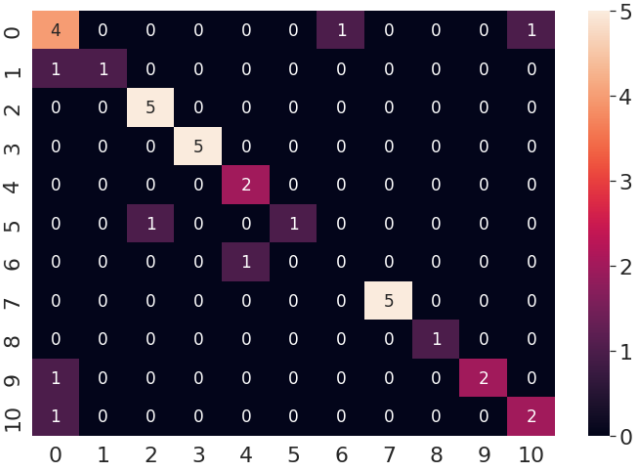


Figura 6.39: Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA

Classes	***Precision***	***Recall***	***F1-Score***
0	*57%*	*67%*	*62%*
1	*100%*	*50%*	*67%*
2	*83%*	*100%*	*91%*
3	*100%*	*100%*	*100%*
4	*67%*	*100%*	*80%*
5	*100%*	*50%*	*67%*
6	*0%*	*0%*	*0%*
7	*100%*	*100%*	*100%*
8	*100%*	*100%*	*100%*
9	*100%*	*67%*	*80%*
10	*67%*	*67%*	*67%*
Accuracy		*80%*	

Tabela 6.24: Predição com as métricas da matriz de confusão com a base de dados 11_Tumores

Para o algoritmo RF é detalhado o desempenho por classe por ser o experimento desta base que obteve a menor performance. Nota-se na Tabela 6.24 que a classe 6 obteve 0%. Esta classe é vinda da amostra de câncer de fígado que possui apenas 7 exemplos. Como foi usado 20% do dado para o teste, apenas um exemplo foi testado para a classe 6 e se confundiu com a classe 4 que é o câncer de gastroesôfago e possui 2 exemplos (2 exemplos é 20% de um total de 12 exemplos na base de dados 11_Tumores). Esta base de dados tem detalhes interessantes por ser de multicitasse e classes desbalanceadas. Podemos analisar pela figura 6.39 e pegar como exemplo para avaliar outras matrizes de confusão deste experimento. Por exemplo, podemos identificar os falsos positivos e falsos negativos.

Vamos tomar como exemplo a classe 0: A classe 0 possui 3 exemplos classificados errados como falso positivo que são as classes 1, 9, 10 e que se classificaram como sendo da

classe 0; Possui 2 exemplos como falso negativo, esses dois exemplos eram para ser classificados como da classe 0, mais se classificaram como sendo das classes 6 e 10.

Por haver erros tanto para falso positivo quanto para falso negativo. A classe 0 tem 62% de Precisão, 67% de Recall e 62% de F1-Score. Estes percentuais é abaixo em comparação com outras classes pelo fato de que a classe 0 possui em sua classificação falsos positivos e falsos negativos.

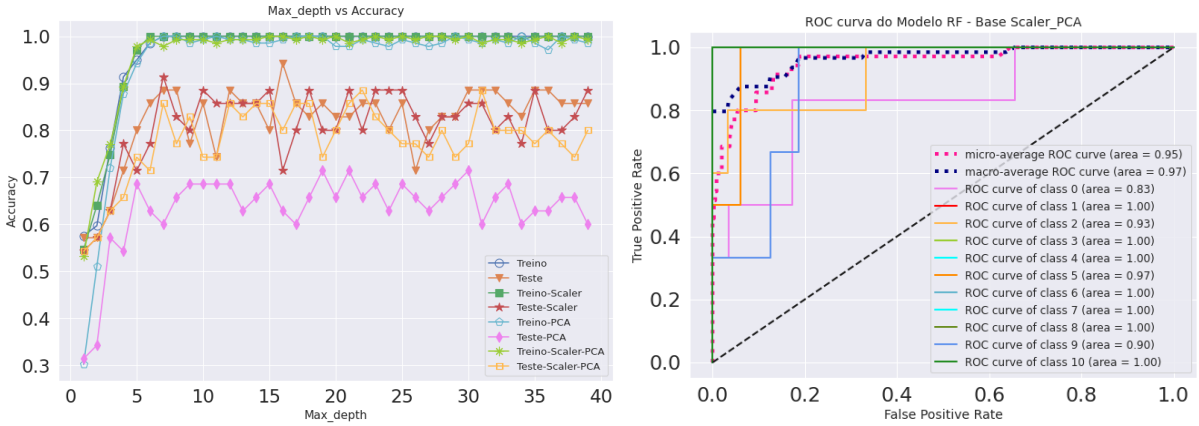


Figura 6.40: À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes

E na Figura 6.40 mostra que o teste PCA obteve o pior desempenho.

• *Compararação dos Algoritmos de ML com a abordagem Nested Cross-validation estratificada*

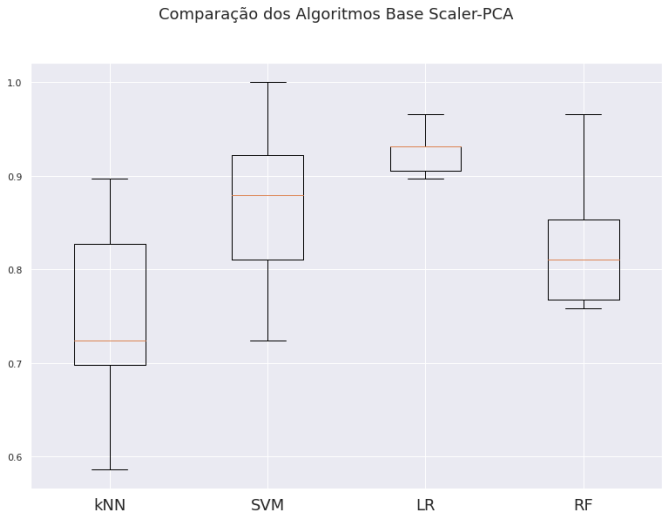


Figura 6.41: Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA

KNN	*Neighbors:10*	*74%*
SVM	*C: 1.0* *kernel: linear*	*86%*
RL	*C : 1* *penalty: l1* *max_iter: 5000*	*92%*
RF	*n_jobs: -1* *n_estimators: 200* *min_samples_split: 10* *min_samples_leaf: 1* *max_features: auto* *max_depth: 40* *criterion: gini* *bootstrap: False*	*82%*

Tabela 6.25: Scores dos algoritmos e seus respectivos parâmetros na base de dados 11_Tumores

Com a validação cruzada Nested Cross-validation que é particionada com 5 *fold* do laço externo e 2 *fold* do laço interno se compara o desempenho dos algoritmos. Nesta avaliação a RL obteve o melhor desempenho com score de 92% superando o método holdout com 85%.

• *Resultado do Algoritmo MLP*

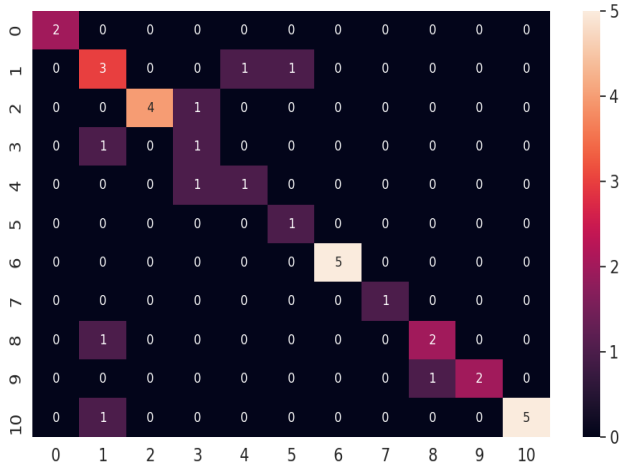


Figura 6.42: Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.86*
F1-Score	*0.82*
Recall	*0.84*
Precision	*0.83*

Tabela 6.26: Predição com as métricas da matriz de confusão com a base de dados 11_Tumores

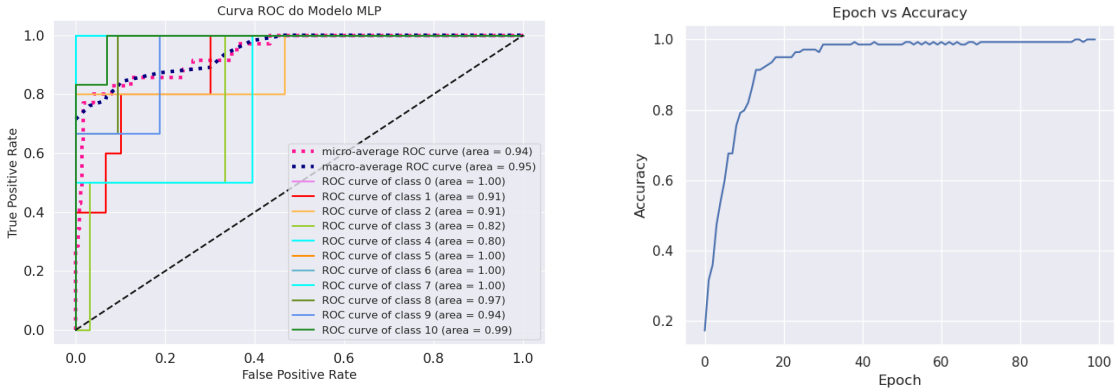


Figura 6.43: À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta.

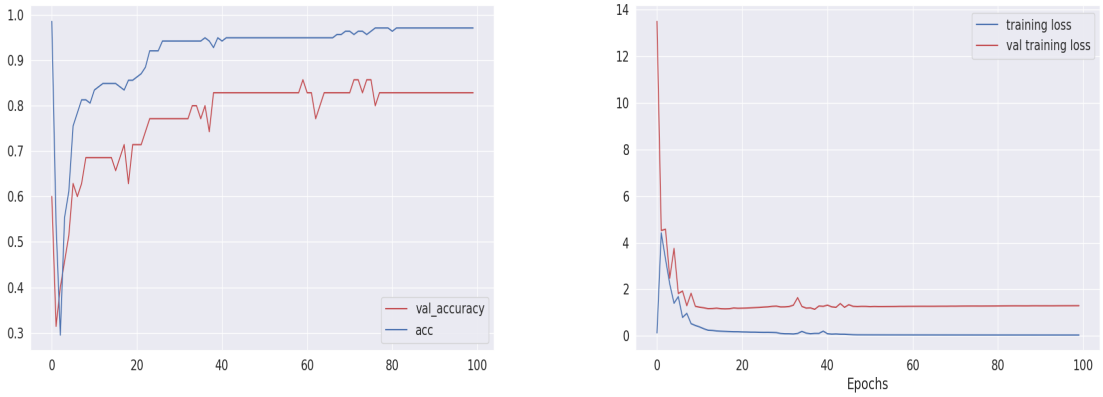


Figura 6.44: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

- *Resultado do Algoritmo CNN*

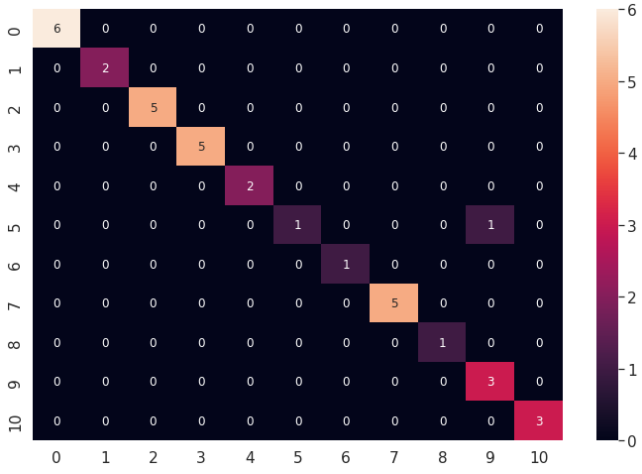


Figura 6.45: Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.97*
F1-Score	*0.96*
Recall	*0.95*
Precision	*0.98*

Tabela 6.27: Predição com as métricas da matriz de confusão com a base de dados 11_Tumores

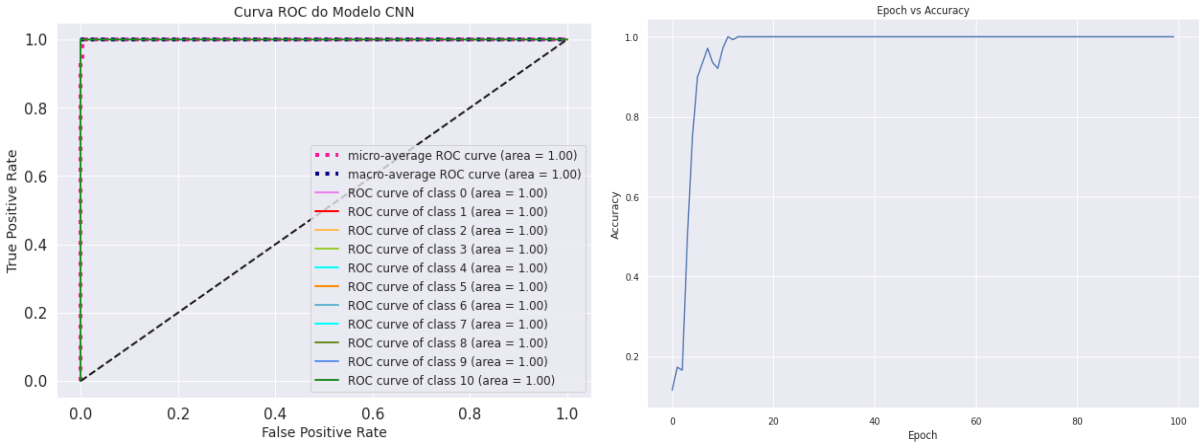


Figura 6.46: À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta.

Figura 6.46 mostra curvas ROC para cada classe usando o método um contra todos para simular um problema de classificação binária. Observe que, como o classificador CNN é tão bom, a AUC é calculada para cada classe e também mostra um modelo geral de bom desempenho.

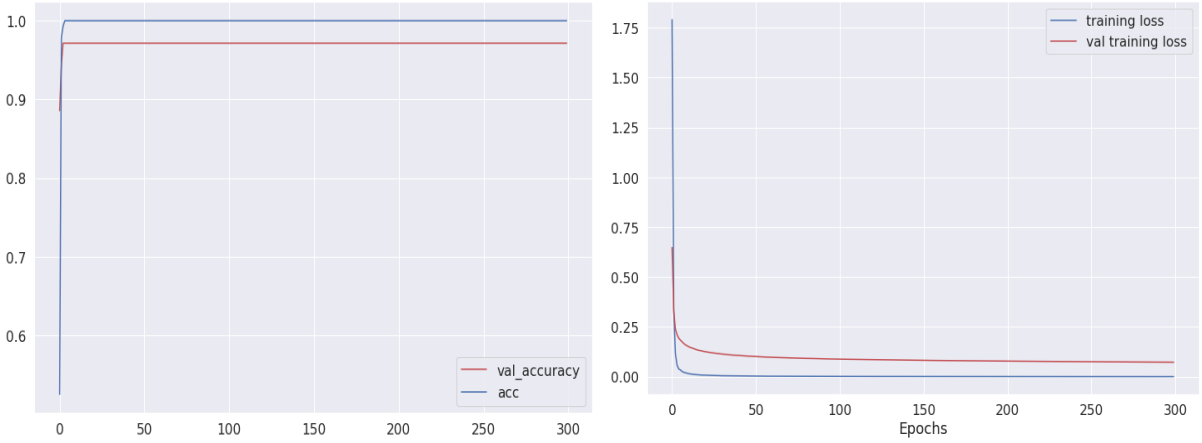


Figura 6.47: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

O algoritmo CNN para a base 11_Tumores obteve o melhor resultado dos algoritmos no método *holdout* com 97%. Na Figura 6.44 do MLP mostra os gráficos de precisão e perda e indica um pouco de lacuna, mais não significativa, e na Figura 6.47 da CNN, mostram que a precisão e a perda estão com suas linhas de treinamento e validação quase juntas indicando um bom ajuste do modelo. Quando ambos os conjuntos de dados mostram resultados muito distantes, a arquitetura pode estar superdimensionada onde pode ocorrer a indicação de overffiting.

Parâmetros	***MLP***	***CNN***
Batch size	*20*	*10*
Epochs	*100*	*10*
Training optimization algorithm	*Adam*	*SGD*
Learning rate	*0,2*	*0,1*
Momentum	*N/A*	*0,0*
Network weight initialization	*N/A*	*Glorot_normal*
Neuron activation function	*relu*	*linear*
Dropout regularization	*0,0*	*0,5*
Weight constraint	*N/A*	*1*
Number of neurons in the hidden layers	*50*	*100*
Filters	*N/A*	*128*
Kernel Size	*N/A*	*3*

Tabela 6.28: Parâmetros dos algoritmos das Redes Neurais para a base de dados 11_Tumores

• *Comparação dos Algoritmos de ML e DL com a Validação Cruzada Estratificada*

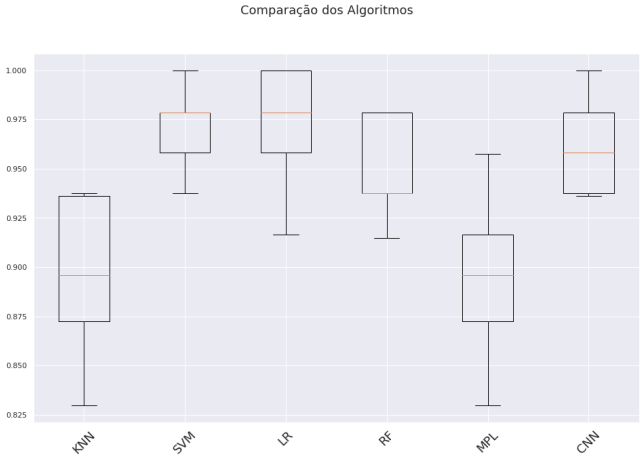


Figura 6.48: Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA

KNN	*0.89*
SVM	*0.97*
RL	*0.97*
RF	*0.94*
MLP	*0.89*
CNN	*0.96*

Tabela 6.29: Scores dos algoritmos com o método validação cruzada estratificada na base de dados 11_Tumores

Para validação cruzada estratificada na base de dados 11_Tumores todos os algoritmos obtiveram uma boa performance.

Os experimentos desta base de dados são similares ao do artigo (Tabares-Soto et al., 2020), que conseguiu melhores resultados com as redes neurais DL.

Modelo do Pré-processamento	***KNN***	***SVM***	***RL***	***RF***	***MLP***	***CNN***
Base Original	*91%*	*91%*	*97%*	*97%*	*86%*	*96%*
Base Scaler	*85%*	*88%*	*94%*	*97%*	*82%*	*89%*
Base PCA	*85%*	*94%*	*97%*	*91%*	*84%*	*80%*
Base PCA Scaler	*82%*	*88%*	*94%*	*82%*	*83%*	*98%*

Tabela 6.30: Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados 11_Tumores

Em geral todos os algoritmos obtiveram um bom desempenho independente dos diferentes testes de pré-processamento. Os algoritmos de RL e CNN foram os melhores com destaque para CNN que atingiu 98% no método *holdout*.

6.4 Algoritmos com a base de dados Leucemia

Esta base contém 248 amostras e os algoritmos usam 20% para o teste e assim restando 50 amostras para avaliar o desempenho da classificação.

- *Resultado do Algoritmo KNN*

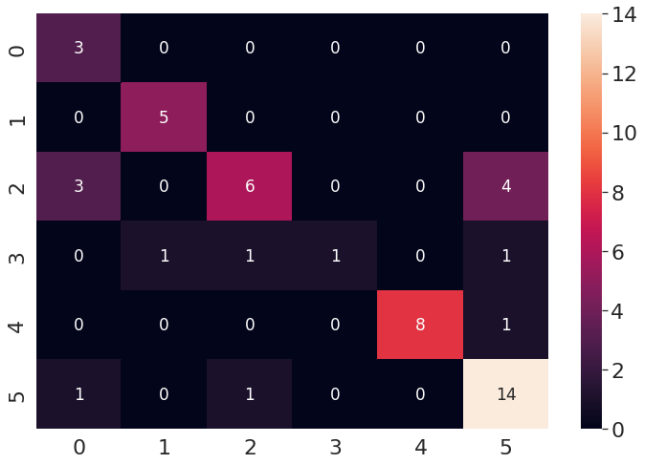


Figura 6.49: Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA

Classes	***Precision***	***Recall***	***F1-Score***
0	*43%*	*100%*	*60%*
1	*83%*	*100%*	*91%*
2	*75%*	*46%*	*57%*
3	*100%*	*25%*	*40%*
4	*100%*	*89%*	*94%*
5	*70%*	*88%*	*78%*
Accuracy	*74%*		

Tabela 6.31: Predição com as métricas da matriz de confusão com a base de dados Leucemia

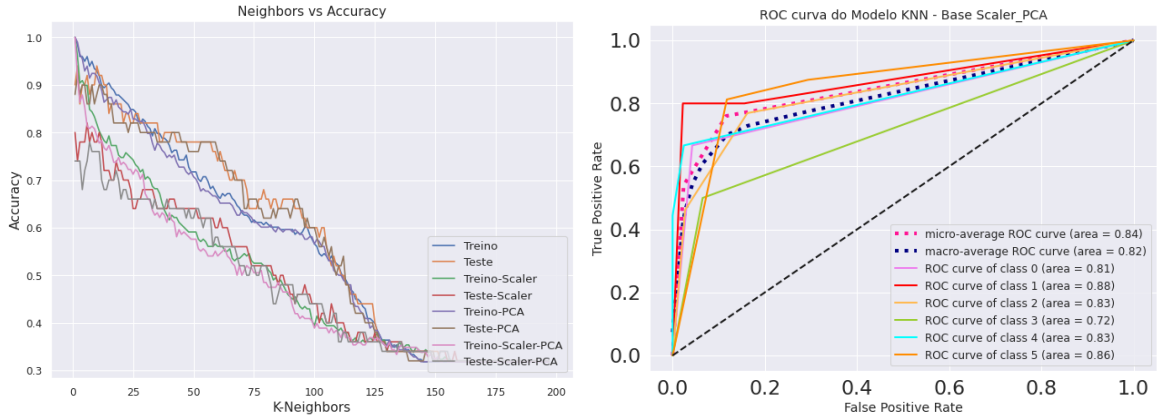


Figura 6.50: À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes

Na figura 6.49 matriz de confusão é referente ao K=7. O KNN não mostra uma boa classificação e na Tabela 6.31 podemos ver o desempenho de cada classe. Esta base contém 6 subtipos de leucemia e a classe 2 se confundiu com a classe 0 e com a classe 5 que possui a maior quantidade de exemplo desta base.

A métrica recall ou taxa de positivo verdadeiro atribuiu 46% para as classe 2 e 25% para a classe 3, o que significa a porcentagem de uma determinada classe identificada corretamente (de todos os exemplos dados dessa classe).

A métrica F1-Score atribuiu 60% para a classe 0 sendo que seu acerto ou taxa de positivo verdadeiro foi de 100%, isto se deve pelo fato das classes 2 e 5 terem se classificado como a classe 0 caracterizando como falsos positivos. As classes 2 e 3 obtiveram valores baixos para esta métrica 57% e 40% respectivamente. A classe 2 por ter tanto falsos positivos e negativos e a classe 3 por ter falsos negativos.

- *Resultado do Algoritmo SVM*

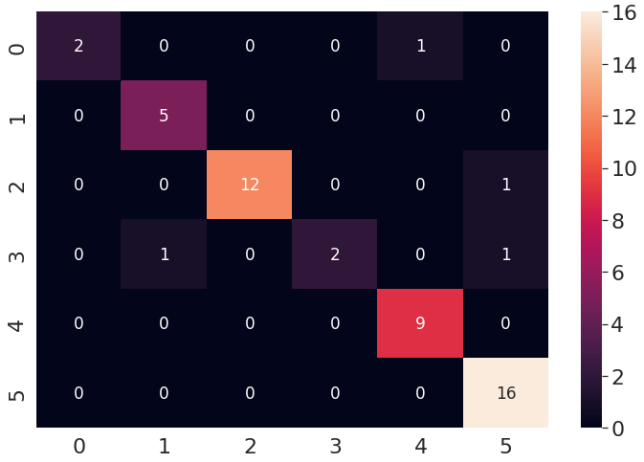


Figura 6.51: Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.92*
F1-Score	*0.91*
Recall	*0.92*
Precision	*0.92*

Tabela 6.32: Predição com as métricas da matriz de confusão com a base de dados Leucemia

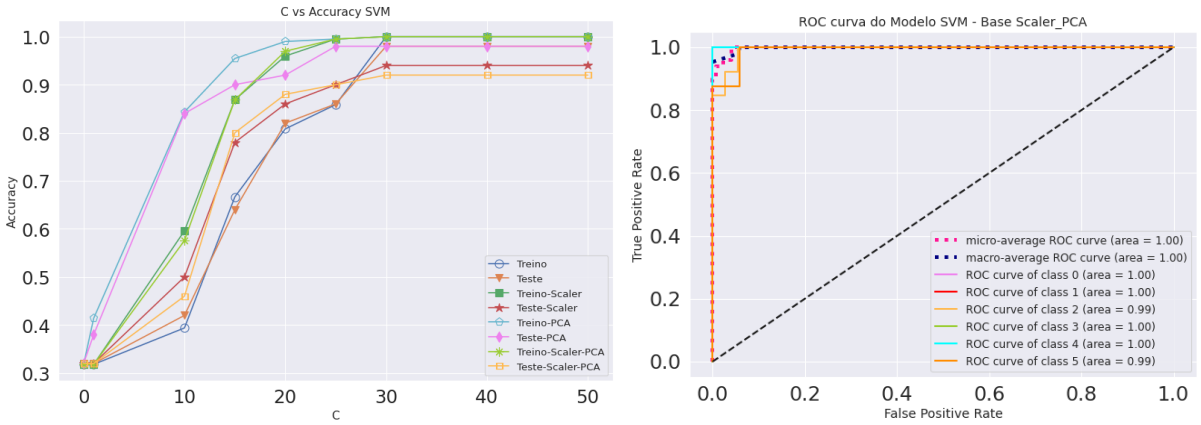
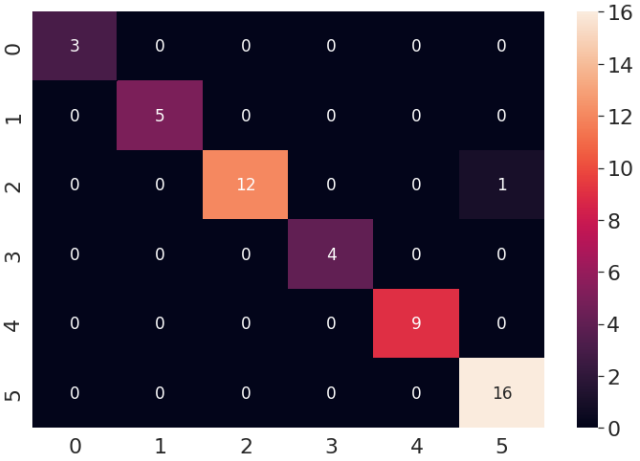


Figura 6.52: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

- *Resultado do Algoritmo RL*



Accuracy	*0.98*
F1-Score	*0.97*
Recall	*0.98*
Precision	*0.98*

Tabela 6.33: Predição com as métricas da matriz de confusão com a base de dados Leucemia

Figura 6.53: Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA

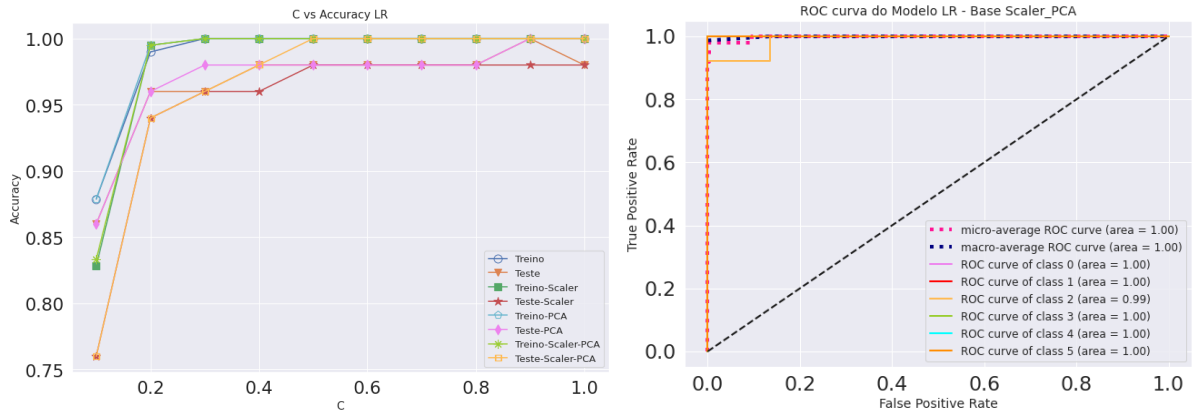


Figura 6.54: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

A matriz de confusão do SVM Figura 6.51 refere-se ao parâmetro C=30 onde a base teste Scaler PCA atinge este valor como é visto na Figura 6.52.

• *Resultado do Algoritmo RF*

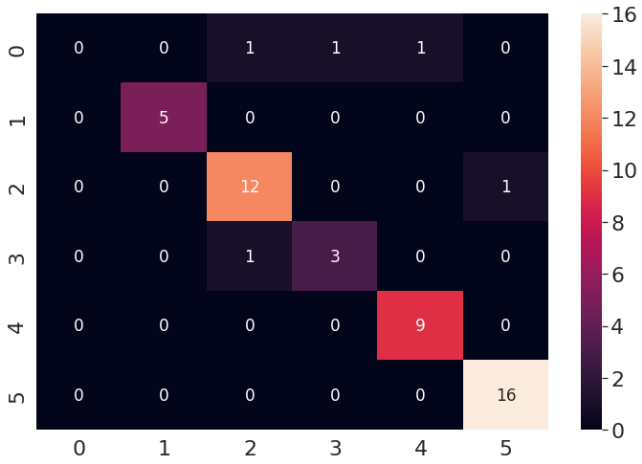


Figura 6.55: Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.90*
F1-Score	*0.87*
Recall	*0.90*
Precision	*0.84*

Tabela 6.34: Predição com as métricas da matriz de confusão com a base de dados Leucemia

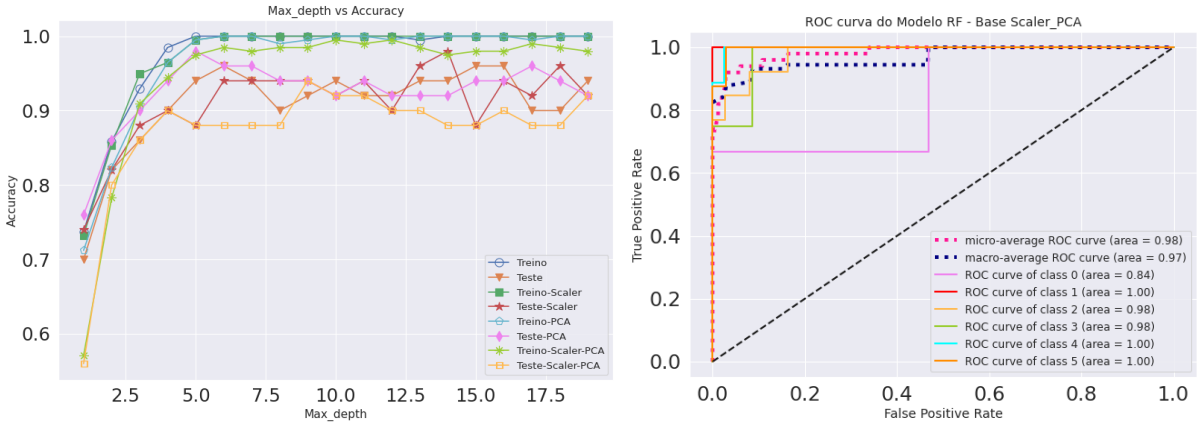


Figura 6.56: À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes

Os algoritmos RL para base de dados Leucemia apresentam uma boa generalização do modelo. Visto que na Figura 6.53 da matriz de confusão da RL o classificador errou somente a classe 2 para uma amostra. E o RF obteve um desempenho melhor nesta base de dados, embora tenha errado todas as amostras da classe 0 como é visto na matriz de confusão da Figura 6.55.

• *Compararação dos Algoritmos de ML com a abordagem Nested Cross-validation estratificada*

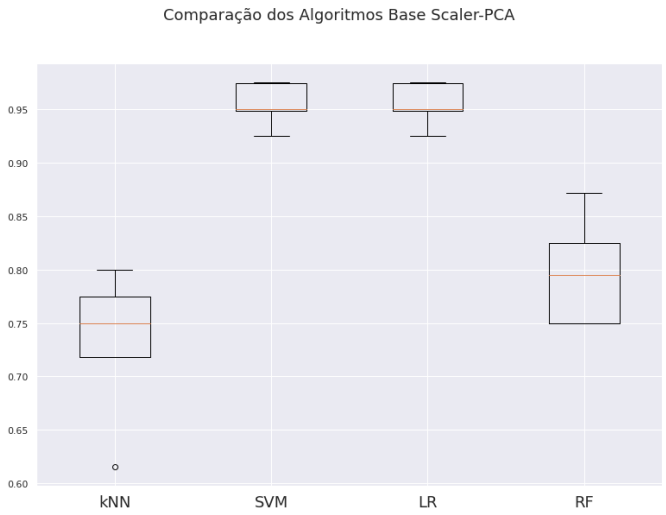


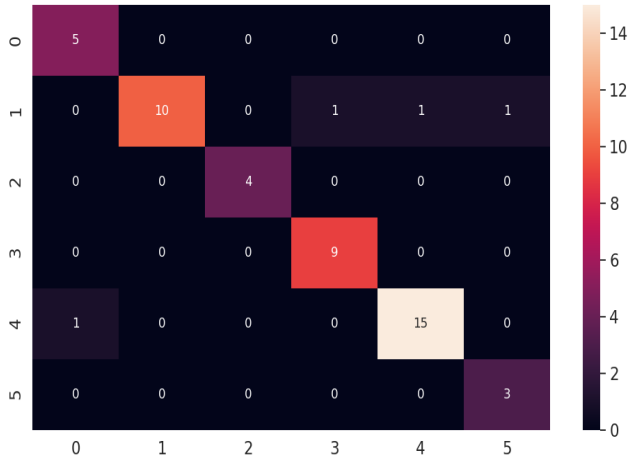
Figura 6.57: Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA

KNN	*Neighbors:3*	*73%*
SVM	*C: 1.0* *kernel: linear*	*95%*
RL	*C : 5* *penalty: l1* *max_iter: 5000*	*95%*
RF	*n_jobs: -1* *n_estimators: 400* *min_samples_split: 2* *min_samples_leaf: 1* *max_features: auto* *max_depth: 30* *criterion: gini* *bootstrap: False*	*70%*

Tabela 6.35: Scores dos algoritmos e seus respectivos parâmetros na base de dados Leucemia

Com a validação cruzada Nested Cross-validation com 5 *fold* do laço externo e 2 *fold* do laço interno é comparado o desempenho dos algoritmos. Nesta avaliação a SVM e RL obteve o melhor desempenho com score de 95% cada.

• *Resultado do Algoritmo MLP*



Accuracy	*0.92*
F1-Score	*0.92*
Recall	*0.95*
Precision	*0.90*

Tabela 6.36: Predição com as métricas da matriz de confusão com a base de dados Leucemia

Figura 6.58: Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA

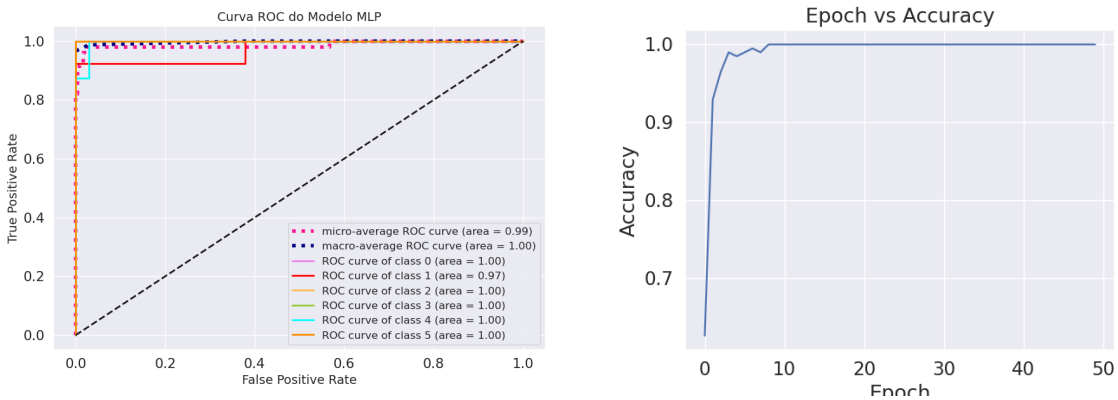


Figura 6.59: À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta.

Figura 6.59 mostra curvas ROC para cada classe usando o método um contra todos para simular um problema de classificação binária. Observe que, como o classificador MLP é tão bom, as linhas que representam as classes estão próximas no canto superior esquerdo. A AUC é calculada para cada classe e também mostra um modelo geral de bom desempenho.

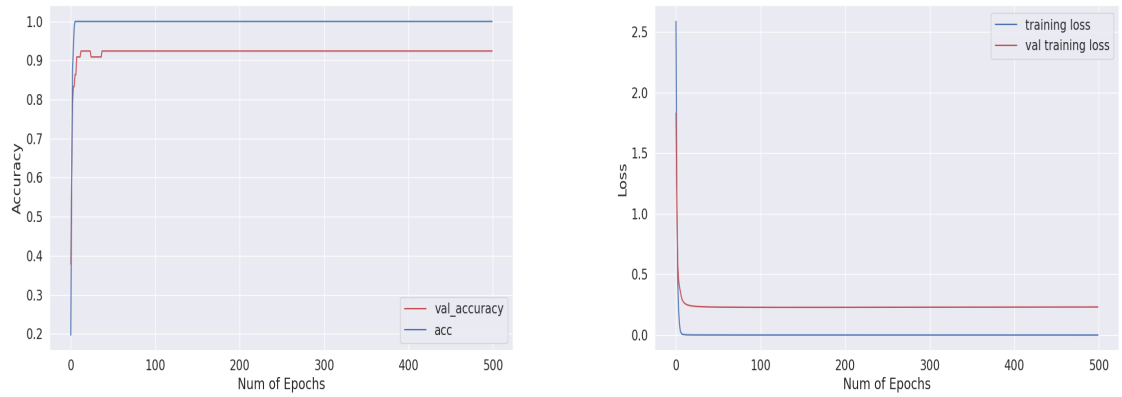


Figura 6.60: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

- *Resultado do Algoritmo CNN*

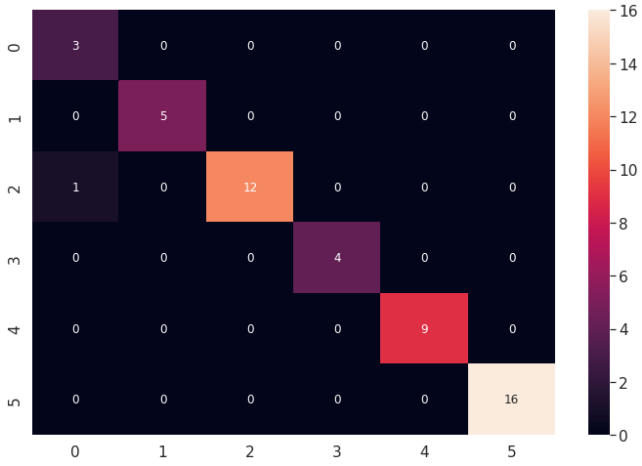


Figura 6.61: Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.98*
F1-Score	*0.97*
Recall	*0.99*
Precision	*0.96*

Tabela 6.37: Predição com as métricas da matriz de confusão com a base de dados Leucemia

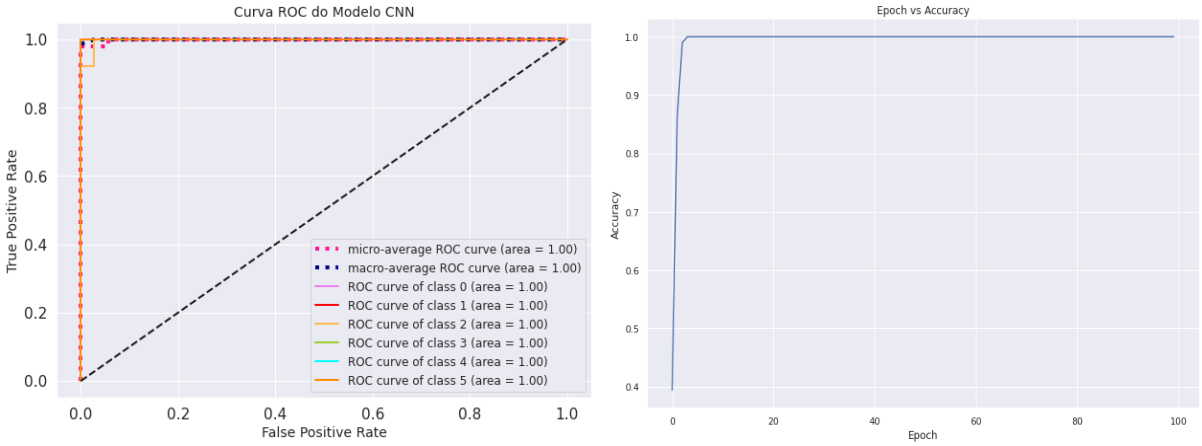


Figura 6.62: À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta.

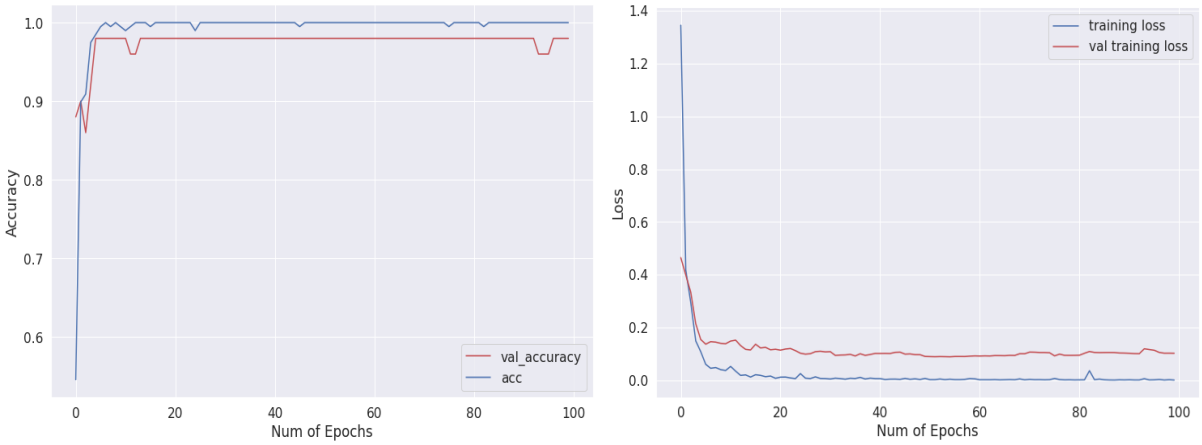


Figura 6.63: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

O algoritmo CNN para a base Leucemia obteve novamente um bom desempenho no método *holdout* com 98% de *accuracy*. Em sua matriz de confusão da Figura 6.61 mostra somente um erro da amostra da classe 2.

Na Figura 6.60 e na Figura 6.63 do MLP e CNN respectivamente os gráficos mostram que a precisão e a perda estão com suas linhas de treinamento e validação quase juntas indicando um bom ajuste do modelo. Quando ambos os conjuntos de dados mostram resultados muito distantes, a arquitetura pode estar superdimensionada onde pode ocorrer a indicação de overffiting.

Parâmetros	***MLP***	***CNN***
Batch size	*10*	*10*
Epochs	*100*	*100*
Training optimization algorithm	*Adam*	*SGD*
Learning rate	*0,1*	*0,01*
Momentum	*N/A*	*0.8*
Network weight initialization	*N/A*	*lecun_uniform*
Neuron activation function	*relu*	*linear*
Dropout regularization	*0,5*	*0,5*
Weight constraint	*N/A*	*1*
Number of neurons in the hidden layers	*50*	*100*
Filters	*N/A*	*128*
Kernel Size	*N/A*	*3*

Tabela 6.38: Parâmetros dos algoritmos das Redes Neurais para a base de dados Leucemia

• *Comparação dos Algoritmos de ML e DL com a Validação Cruzada Estratificada*

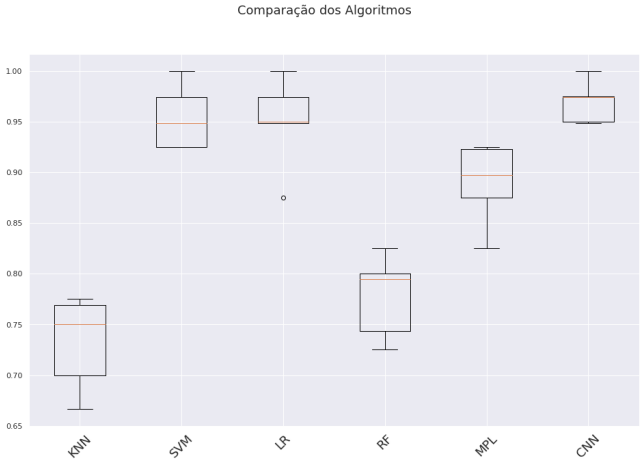


Figura 6.64: Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA

KNN	*0.73*
SVM	*0.95*
RL	*0.94*
RF	*0.77*
MLP	*0.88*
CNN	*0.96*

Tabela 6.39: Scores dos algoritmos com o método validação cruzada estratificada na base de dados Leucemia

Para validação cruzada estratificada na base de dados Leucemia os algoritmos SVM, RL e CNN são os que atingiram os melhores scores.

Modelo do Pré-processamento	***KNN***	***SVM***	***RL***	***RF***	***MLP***	***CNN***
Base Original	*94%*	*98%*	*100%*	*96%*	*95%*	*100%*
Base Scaler	*82%*	*94%*	*98%*	*98%*	*98%*	*98%*
Base PCA	*92%*	*98%*	*100%*	*98%*	*97%*	*100%*
Base PCA Scaler	*78%*	*92%*	*100%*	*94%*	*92%*	*98%*

Tabela 6.40: Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados Leucemia

Em geral todos os algoritmos obtiveram um bom desempenho independente dos diferentes testes de pré-processamento. Os algoritmos de RL e CNN foram os melhores para o método *holdout*.

6.5 Algoritmos com a base de dados SRBCT

Esta base contém 63 amostras e os algoritmos usam 20% para o teste e assim restando 13 amostras para avaliar o desempenho da classificação.

Em outros estudos o desempenho da base de dados SRBCT atinge uma accuracy de 100% de modo geral como é visto em (GÜÇKIRAN et al., 2019). O desafio é alcançar o mesmo desempenho.

- *Resultado do Algoritmo KNN*

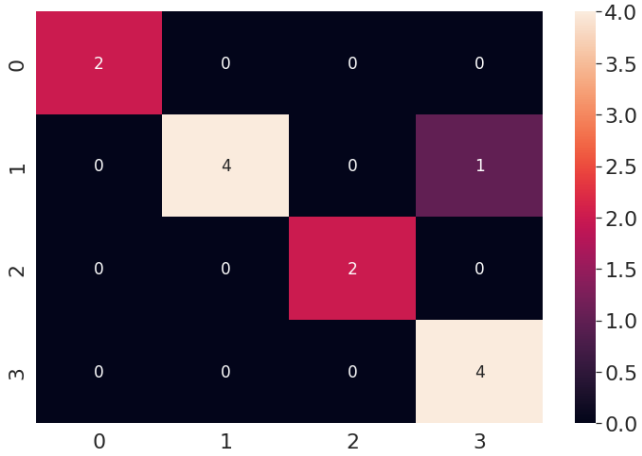


Figura 6.65: Algoritmo KNN e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*0.92*
F1-Score	*0.92*
Recall	*0.92*
Precision	*0.93*

Tabela 6.41: Predição com as métricas da matriz de confusão com a base de dados SRBCT

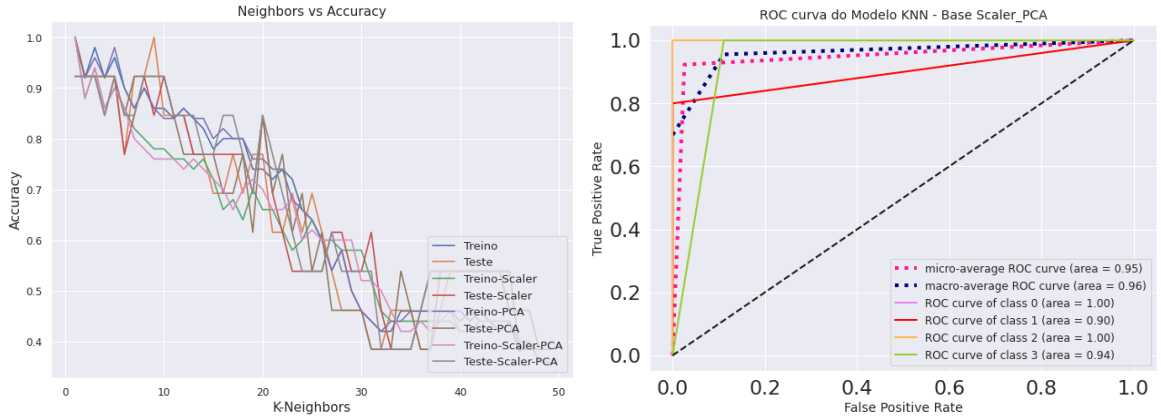


Figura 6.66: À esquerda, o comportamento das bases a medida que aumenta os K vizinhos, à direita a área da curva ROC para cada uma das classes

- *Resultado do Algoritmo SVM*

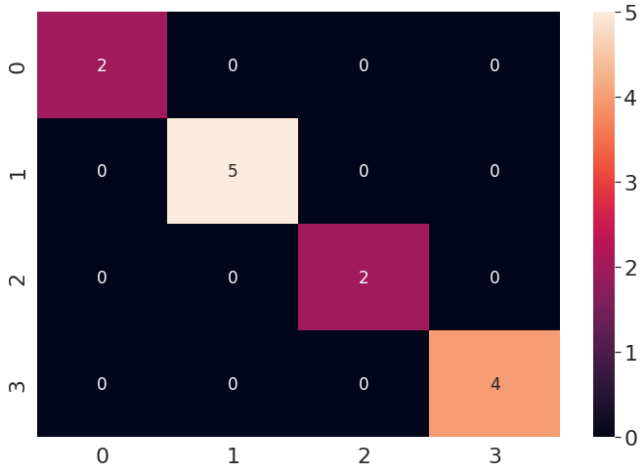


Figura 6.67: Algoritmo SVM e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*100*
F1-Score	*100*
Recall	*100*
Precision	*100*

Tabela 6.42: Predição com as métricas da matriz de confusão com a base de dados SRBCT

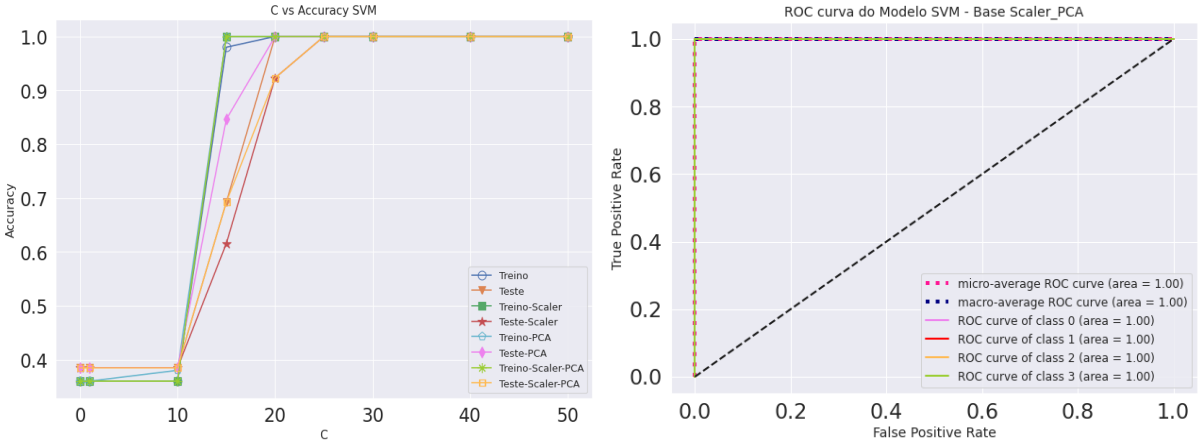


Figura 6.68: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

- *Resultado do Algoritmo RL*

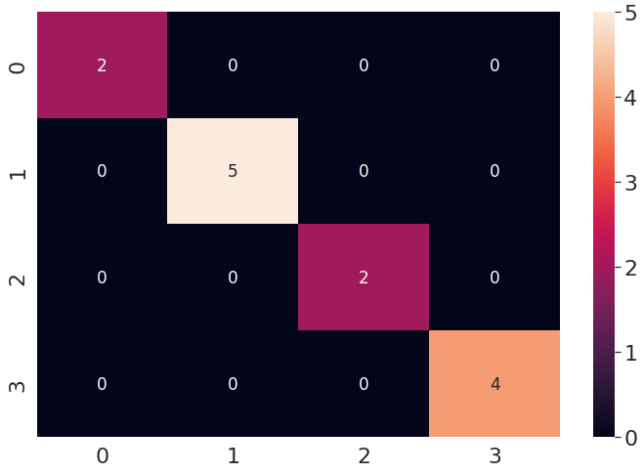


Figura 6.69: Algoritmo RF e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*100*
F1-Score	*100*
Recall	*100*
Precision	*100*

Tabela 6.43: Predição com as métricas da matriz de confusão com a base de dados SRBCT

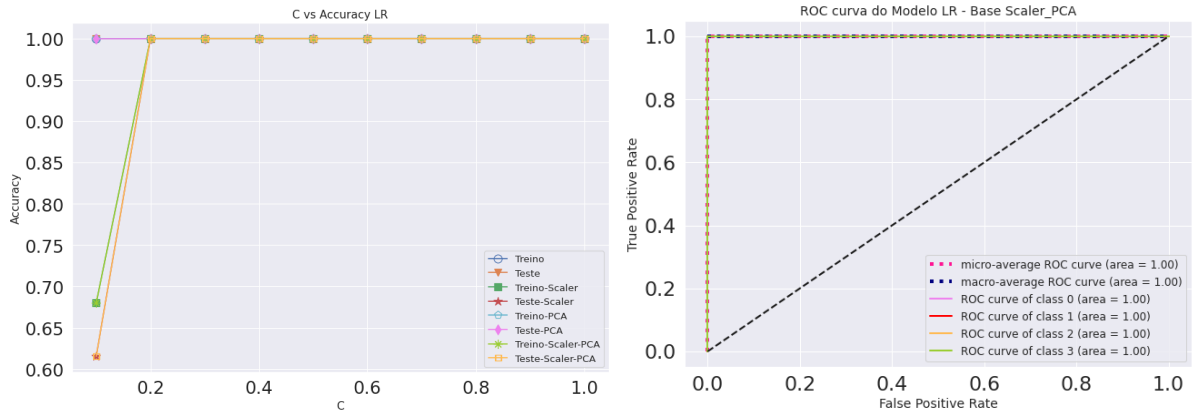


Figura 6.70: À esquerda, o comportamento das bases a medida que aumenta a profundidade da árvore, à direita a área da curva ROC para cada uma das classes

De um modo geral os algoritmos SVM e RL atingiram uma predição de 100% tando na matriz de confusão como na curva ROC. O KNN obteve uma boa pontuação errando um exemplo da classe 1.

- *Resultado do Algoritmo RF*

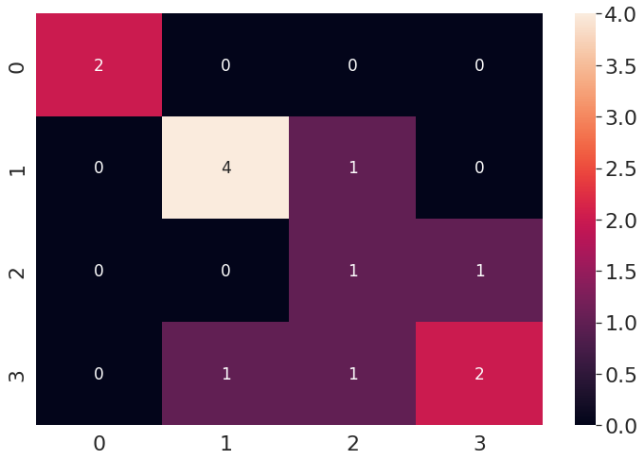


Figura 6.71: Algoritmo RL e a matriz de confusão com o método de pré-processamento Scaler PCA

Classes	***Precision***	***Recall***	***F1-Score***
0	*100%*	*100%*	*100%*
1	*80%*	*80%*	*80%*
2	*33%*	*50%*	*40%*
3	*67%*	*50%*	*57%*
Accuracy		*69%*	

Tabela 6.44: Predição com as métricas da matriz de confusão com a base de dados SRBCT

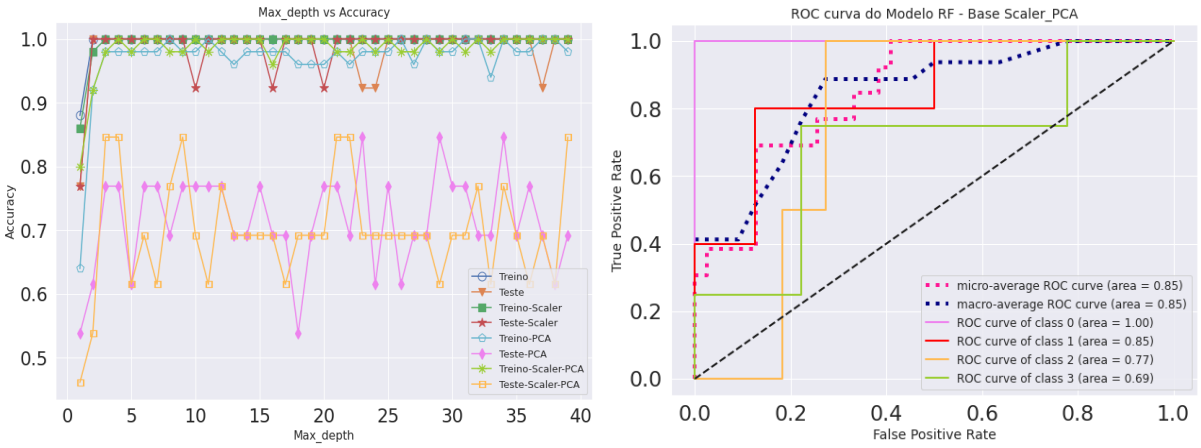


Figura 6.72: À esquerda, o comportamento das bases a medida que aumenta o valor de C, à direita a área da curva ROC para cada uma das classes

O algoritmo RF, obteve um desempenho diferente neste conjunto de dados em comparação com o KNN, RL e SVM. Este fato se repete em conjuntos de dados anteriores deste experimento, demostrando ser menos eficiente nas bases de teste Scaler PCA. Isto demostra que o algoritmo RF não se ajustou com os parâmetros escolhido neste experimento.

• *Comparação dos Algoritmos de ML com a abordagem Nested Cross-validation estratificada*

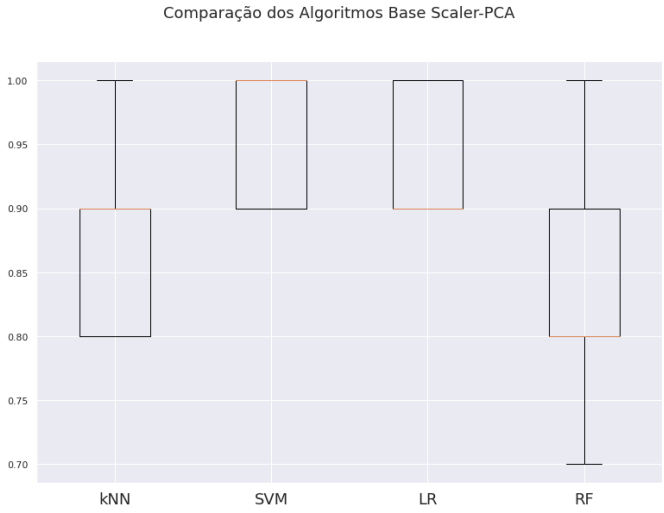


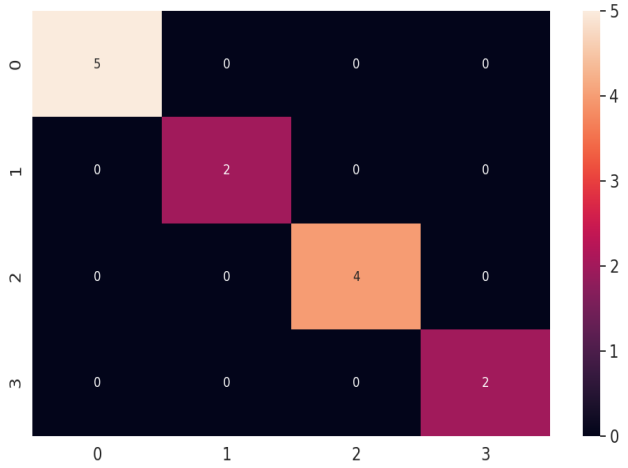
Figura 6.73: Desempenho e comparação dos classificadores de ML com o método Nested Cross-validation nos dados de pré-processamento Scaler PCA

KNN	*Neighbors:6*	*88%*
SVM	*C: 0.01* *kernel: linear*	*96%*
RL	*C : 10* *penalty: l1* *max_iter: 500*	*94%*
RF	*n_jobs: -1* *n_estimators: 800* *min_samples_split: 5* *min_samples_leaf: 2* *max_features: auto* *max_depth: 40* *criterion: gini* *bootstrap: False*	*84%*

Tabela 6.45: Scores dos algoritmos e seus respectivos parâmetros na base de dados SRBCT

Com a validação cruzada Nested Cross-validation onde a base é particionada com 5 fold do laço externo e 2 fold do laço interno é comparado o desempenho dos algoritmos. Este método estima a real performance e busca pelos melhores parâmetros e fez com que o algoritmo RF atingisse uma melhora, de 69% para 84% na classificação dos dados.

- *Resultado do Algoritmo MLP*



Accuracy	*100*
F1-Score	*100*
Recall	*100*
Precision	*100*

Tabela 6.46: Predição com as métricas da matriz de confusão com a base de dados SRBCT

Figura 6.74: Algoritmo MLP e a matriz de confusão com o método de pré-processamento Scaler PCA

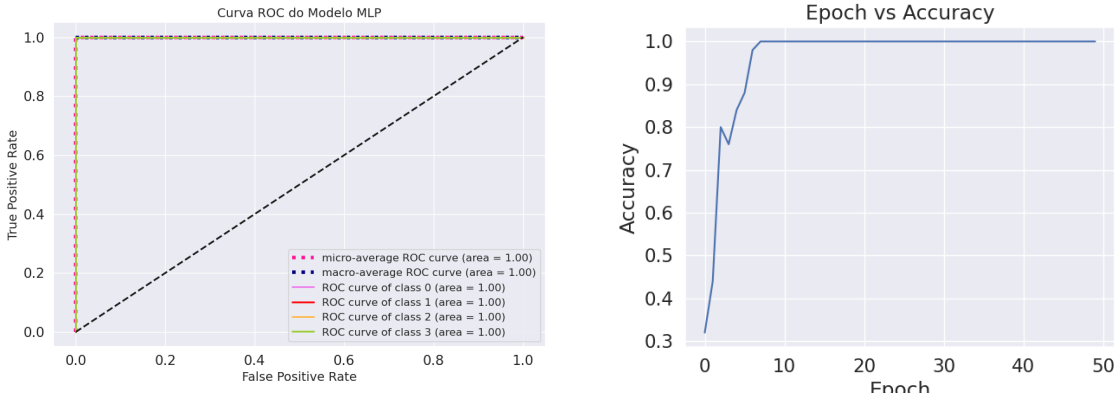


Figura 6.75: À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta.

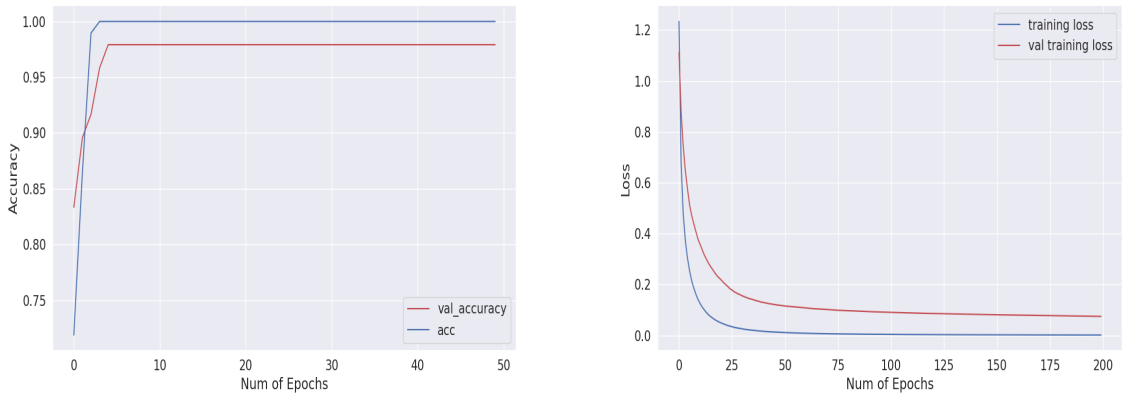


Figura 6.76: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

• *Resultado do Algoritmo CNN*

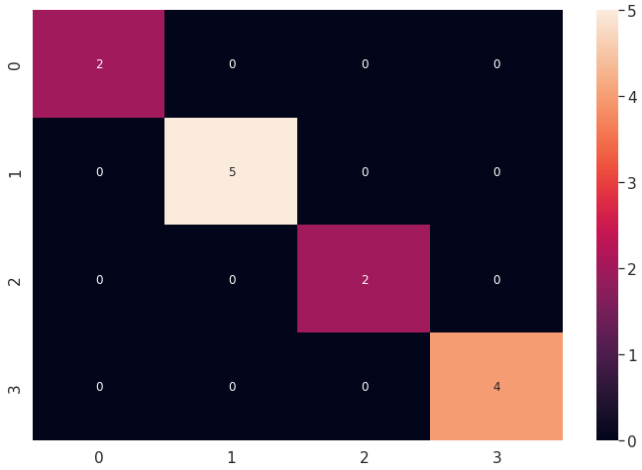


Figura 6.77: Algoritmo CNN e a matriz de confusão com o método de pré-processamento Scaler PCA

Accuracy	*100*
F1-Score	*100*
Recall	*100*
Precision	*100*

Tabela 6.47: Predição com as métricas da matriz de confusão com a base de dados SRBCT

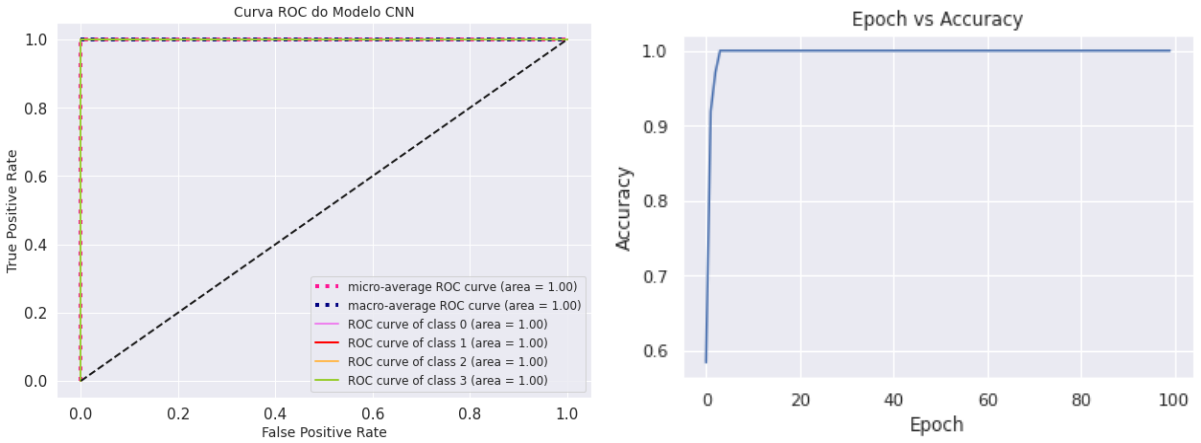


Figura 6.78: À esquerda, a área da curva ROC para cada uma das classes, à direita o comportamento do treinamento à medida que o número de épocas aumenta.

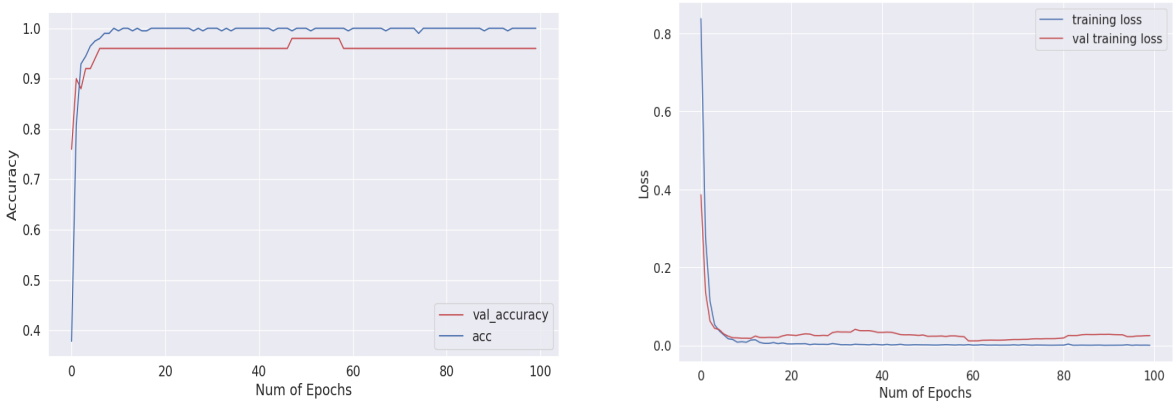


Figura 6.79: À esquerda, o comportamento da accuracy do treinamento e sua validação, à direita a função de perda para o treinamento e validação.

Os algoritmos MLP e CNN para a base SRBCT alcançaram um excelente desempenho com predição de 100% no método *holdout*.

Na Figura 6.76 e 6.79 do MLP e CNN respectivamente os gráficos mostram que a precisão e a perda estão com suas linhas de treinamento e validação quase juntas indicando um bom ajuste do modelo. Quando ambos os conjuntos de dados mostram resultados muito distantes, a arquitetura pode estar superdimensionada onde pode ocorrer a indicação de overffiting.

Parâmetros	***MLP***	***CNN***
Batch size	*10*	*10*
Epochs	*10*	*10*
Training optimization algorithm	*Adam*	*SGD*
Learning rate	*0.1*	*0.2*
Momentum	*N/A*	*0,6*
Network weight initialization	*N/A*	*he_normal*
Neuron activation function	*relu*	*relu*
Dropout regularization	*0,0*	*0,0*
Weight constraint	*N/A*	*1*
Number of neurons in the hidden layers	*50*	*100*
Filters	*N/A*	*32*
Kernel Size	*N/A*	*3*

Tabela 6.48: Parâmetros dos algoritmos das Redes Neurais para a base de dados SRBCT

• *Comparação dos Algoritmos de ML e DL com a Validação Cruzada Estratificada*

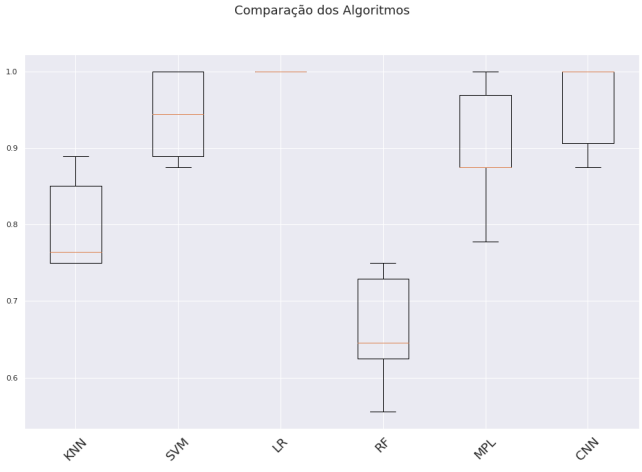


Figura 6.80: Desempenho e comparação dos classificadores de ML e DL nos dados de pré-processamento Scaler PCA

KNN	*0.79*
SVM	*0.94*
RL	*1.00*
RF	*0.66*
MLP	*0.90*
CNN	*0.95*

Tabela 6.49: Scores dos algoritmos com o método validação cruzada estratificada na base de dados SRBCT

Na comparação de todos os algoritmos na validação cruzada estratificada, no conjunto de dados SRBCT, RL se manteve e se destaca como o melhor score de 100%, seguido do SVM e CNN com 94% e 95% respectivamente. O KNN e RF com um score abaixo do método holdout e Nested Cross-validation.

Modelo do Pré-processamento	***KNN***	***SVM***	***RL***	***RF***	***MLP***	***CNN***
Base Original	*100%*	*100%*	*100%*	*100%*	*100%*	*100%*
Base Scaler	*92%*	*100%*	*100%*	*100%*	*100%*	*100%*
Base PCA	*100%*	*100%*	*100%*	*84%*	*97%*	*100%*
Base PCA Scaler	*92%*	*100%*	*100%*	*84%*	*100%*	*100%*

Tabela 6.50: Precisão dos algoritmos com suas respectivas diferenças de pré-processamento para a base de dados SRBCT

Em geral todos os algoritmos obtiveram um bom desempenho independente dos diferentes testes de pré-processamento para o conjunto de dados SRBCT. Os algoritmos de SVM, RL e CNN foram os melhores para o método *holdout* alcançando 100% de accuracy.

6.6 Resumo dos Resultados

- Método Holdout
  - Conjunto de dados Câncer de Pulmão: Em média de todos os algoritmos obtiveram um bom desempenho, as partições Original e Scaler PCA obtiveram melhores resultados do que as demais.
  - Conjunto de dados Câncer de Próstata: A partição de teste Scaler PCA foi a melhor para todos os algoritmos com 90%, com exceção do RF que alcançou 61%.
  - Conjunto de dados 11_Tumores: A partição de teste Original, em média obteve o melhor desempenho, os algoritmos SVM, RL e CNN se destacaram.
  - Conjunto de dados Leucemia: A partição de teste Original, em média obteve o melhor desempenho.
  - Conjunto de dados SRBCT: A partição de teste Original, em média obteve o melhor desempenho.
- Nested Cross-validation estratificada avaliou os algoritmos de ML e demostra um resultado que estima um desempenho de generalização imparcial, com pontuações abaixo do método holdout. Foi integrada a versão estratificada para que todos os exemplos de uma classe minoritária estivesse em todas as partições da validação. O RF obteve uma melhora neste método, pois encontrou melhores parâmetros.
- Validação cruzada estratificada que se propõe evitar estimativas não realistas de desempenho dos classificadores.

  Na média simples de todas as bases, os algoritmos de ML e DL se classificam como:
  - RL : 96%
  - CNN : 95%
  - SVM : 94%
  - MLP : 89%
  - KNN : 82%
  - RF : 81%

Os algoritmos RF e CNN demostraram ter uma boa performance em dados originais com o método holdout e sem o pré-processamento. O algoritmo RF não se ajustou para a base de teste Scale PCA, mas se recuperou na validação cruzada. Como em ambos os métodos da CV proposto para este trabalho, possui a abordagem estratificada, e são obtidos resultados mais confiáveis com relação às bases de dados com classes desbalanceadas.

Na maioria dos experimentos o método *holdout* obteve os melhores resultados. Mas com este método, os dados são divididos aleatoriamente em treino e teste. O que pode ocorrer é que os dados que antes eram utilizados para teste agora podem estar sendo utilizados para treino e vice-versa, este processo pode enviesar o modelo e os resultados podem ser muito otimistas. A CV foi utilizada como um método mais confiável pois evita a variância e aleatoriedade dos dados, e também estima a performance do modelo em dados que nunca viu antes. Portanto, a CV obteve uma visão mais imparcial e realista do modelo.

De modo geral os algoritmos obtiveram excelentes resultados, pode-se supor que haja overfitting, uma vez que a matriz de confusão e curva ROC apresentou um comportamento extremamente bom em alguns resultados obtendo 100% de acuracy; no entanto, a comparação das precisões de treinamento e validação entre os parâmetros usando todo o conjunto de dados pode indicar uma precisão perfeita em ambos os conjuntos de dados de treinamento e validação. Outra suposição para se ter obtido um bom resultado em algumas bases, seria pelo fato das bases de dados de microarray possuírem poucas amostras e os algoritmos se tornarem otimistas e enviesados, mas não se conclui neste estudo. Se isto realmente ocorre, precisaria de mais estudos sobre esta hipótese.

7 Conclusões

Ainda nos dias de hoje estamos longe de alcançar uma cura para o câncer e para o futuro, presume que o câncer se torne a doença mais mortal para os humanos, portanto, quanto mais se investir em estudos para contribuir na identificação de genes que tem a tendência a se expressar como tumor maligno, diagnóstico precoce, prognósticos das lesões cancerígenas, indicativos e caminhos mais eficazes para qual tratamento seguir, ou seja, todos estes fatores são necessários para controlar a doença. As técnicas de ML e DL são ferramentas promissoras nesta área e alcançam resultados satisfatórios para a classificação de diferentes tipos de câncer usando conjuntos de dados complexos, como a do *microarrays*.

A proposta para este trabalho se refere a realizar uma ampla comparação experimental entre os algoritmos DL e ML para a classificação e para validar o método proposto. Cinco conjuntos de dados de *microarray* padrão, nomeadamente Câncer de Pulmão, Câncer de Próstata, 11_Tumores, Leucemia, SRBCT são usados. Embora a abordagem para dados de *microarray* tenha alta dimensionalidade e pequena quantidade de exemplos, uma vez que a maioria dos conjuntos de dados têm cerca de 100 amostras, os algoritmos deste estudo se comportaram bem na maioria dos testes, conseguindo uma boa convergência e alcançando boas previsões dos classificadores com acertos de até 100%. Para superar o desafio da alta dimensionalidade e outros problemas associados à natureza dos dados, o PCA é utilizado como técnica de redução de dimensionalidade e o dimensionamento dos atributos é realizado usando a abordagem de normalização *Mini_Max_Scaler*.

Os resultados da CV na maioria dos experimentos demostraram um desempenho pouco abaixo do método *holdout*, mas como a CV fornece uma estimativa mais realista e mais confiável do modelo, neste sentido, os seus resultados ainda foram bons.

Algumas das bases de dados deste trabalho possuem um desequilíbrio de classes, ou seja, somente a base de dados Câncer de Próstata possui seus exemplos de classe balanceada. E para enfrentar o problema de desequilíbrio de classe e obter um resultado da generalização mais confiável e de forma imparcial, a validação cruzada estratificada foi bem sucedida para os conjuntos de dados desbalanceados.

Os algoritmos CNN, SVM e RL se comportaram com melhor desempenho na maioria dos experimentos. Em contrapartida, os algoritmos de KNN, MLP e RF não se ajustaram perfeitamente aos parâmetros. As bases de dados originais obtiveram bom desempenho na maioria dos testes, mas como são de dados ruidosos, os classificadores podem ter uma predição não realística das classes, sendo assim, foram mostrados os gráficos dos resultados da base de teste Scaler PCA que obteve um resultado real do método proposto para este estudo. O algoritmo RF foi muito bem para dados de teste brutos, e sua *accuracy* nos dados de teste com o pré-processamento Scaler PCA foi menor em comparação com os outros algoritmos. Já a CNN se comportou bem para as quatro partições de teste demostrando-se muito bem com dados brutos, e isto é esperado sendo que as convoluções que o algoritmo realiza trabalham para extração das características. Nas etapas deste trabalho foi possível levantar a hipótese que as classificações perfeitas de alguns experimentos se deram pela pouca quantidade de amostras, assim, podendo

enviesar o modelo. Para ter uma melhor conclusão sobre este fator, mais estudos ainda são essenciais para este fim, especialmente quando se trata de conjuntos de dados heterogêneos como conjuntos de dados com classes desbalanceadas e com multiclasse. Em termos de classificação, supervisionado e os métodos ML e DL são considerados de grande interesse nessa direção. Assim, esforço contínuo ainda é necessário para obter métodos de classificação de câncer mais robustos no futuro.

Referências

Aggarwal, C. C. (2017). High-dimensional outlier detection: the subspace method. Em *Outlier analysis*, páginas 149–184. Springer.

Al-Stouhi, S. e Reddy, C. K. (2016). Transfer learning for class imbalance problems with inadequate data. *Knowledge and information systems*, 48(1):201–228.

Alanni, R., Hou, J., Azzawi, H. e Xiang, Y. (2019). A novel gene selection algorithm for cancer classification using microarray datasets. *BMC medical genomics*, 12(1):1–12.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. e Walter, P. (2002). Studying gene expression and function. Em *Molecular Biology of the Cell. 4th edition*. Garland Science.

Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X. et al. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511.

Basavegowda, H. S. e Dagnew, G. (2020). Deep learning approach for microarray cancer data classification. *CAAI Trans. Intell. Technol.*, 5(1):22–33.

Bengio, Y., Courville, A. e Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bolón-Canedo, V., Sánchez-Marono, N., Alonso-Betanzos, A., Benítez, J. M. e Herrera, F. (2014). A review of microarray datasets and applied feature selection methods. *Information Sciences*, 282:111–135.

Brazma, A. e Vilo, J. (2000). Gene expression data analysis. *FEBS letters*, 480(1):17–24.

Cawley, G. C. e Talbot, N. L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, 11:2079–2107.

Ceballos, D., López-Álvarez, D., Isaza, G., Tabares-Soto, R., Orozco-Arias, S. e Ferrin, C. D. (2019). A machine learning-based pipeline for the classification of ctx-m in metagenomics samples. *Processes*, 7(4):235.

Chandra, B. e Gupta, M. (2011). An efficient statistical feature selection approach for classification of gene expression data. *Journal of biomedical informatics*, 44(4):529–535.

Chen, Y., Li, Y., Narayan, R., Subramanian, A. e Xie, X. (2016). Gene expression inference with deep learning. *Bioinformatics*, 32(12):1832–1839.

Cheridito, P., Jentzen, A. e Rossmannek, F. (2021). Non-convergence of stochastic gradient descent in the training of deep neural networks. *Journal of Complexity*, 64:101540.

Chou, E. P. e Ko, T.-W. (2017). Dimension reduction of high-dimensional datasets based on stepwise svm. *arXiv preprint arXiv:1711.03346*.

Dabba, A., Tari, A., Meftali, S. e Mokhtari, R. (2021). Gene selection and classification of microarray data method based on mutual information and moth flame algorithm. *Expert Systems with Applications*, 166:114012.

DeRisi, J., Penland, L., Bittner, M., Meltzer, P., Ray, M., Chen, Y., Su, Y. e Trent, J. (1996). Use of a cdna microarray to analyse gene expression. *Nat. genet*, 14:457–460.

Djebbari, A., Culhane, A. C., Armstrong, A. J. e Quackenbush, J. (2011). Ai methods for analyzing microarray data. Em *Clinical Technologies: Concepts, Methodologies, Tools and Applications*, páginas 877–884. IGI Global.

Eluyode, O. e Akomolafe, D. T. (2013). Comparative study of biological and artificial neural networks. *European Journal of Applied Engineering and Scientific Research*, 2(1):36–46.

Eraslan, G., Avsec, Ž., Gagneur, J. e Theis, F. J. (2019). Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403.

Fakoor, R., Ladhak, F., Nazi, A. e Huber, M. (2013). Using deep learning to enhance cancer diagnosis and classification. Em *Proceedings of the international conference on machine learning*, volume 28, páginas 3937–3949. ACM, New York, USA.

Géron, A. (2019). *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow*. Alta Books.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. e Lander, E. S. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537.

Gordon, G. J., Jensen, R. V., Hsiao, L.-L., Gullans, S. R., Blumenstock, J. E., Ramaswamy, S., Richards, W. G., Sugarbaker, D. J. e Bueno, R. (2002). Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Res*, 62(17):4963–4967.

GÜÇKIRAN, K., Cantürk, İ. e ÖZYILMAZ, L. (2019). Dna microarray gene expression data classification using svm, mlp, and rf with feature selection methods relief and lasso. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 23(1):126–132.

Guillen, P. e Ebalunode, J. (2016). Cancer classification based on microarray gene expression data using deep learning. Em *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, páginas 1403–1405. IEEE.

Guyon, I., Gunn, S., Nikravesh, M. e Zadeh, L. A. (2008). *Feature extraction: foundations and applications*, volume 207. Springer.

Ibrahim, R., Yousri, N. A., Ismail, M. A. e El-Makky, N. M. (2014). Multi-level gene/mirna feature selection using deep belief nets and active learning. Em *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, páginas 3957–3960. IEEE.

Irizarry, R. A., Bolstad, B. M., Collin, F., Cope, L. M., Hobbs, B. e Speed, T. P. (2003). Summaries of affymetrix genechip probe level data. *Nucleic acids research*, 31(4):e15–e15.

Kavitha, K., Ram, A. V., Anandu, S., Karthik, S., Kailas, S. e Arjun, N. (2018). Pca-based gene selection for cancer classification. Em *2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, páginas 1–4. IEEE.

Khan, J., Wei, J. S., Ringner, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C. et al. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673–679.

Kira, K. e Rendell, L. A. (1992). A practical approach to feature selection. Em *Machine learning proceedings 1992*, páginas 249–256. Elsevier.

Krief, S., Faivre, J.-F., Robert, P., Le Douarin, B., Brument-Larignon, N., Lefrere, I., Bouzyk, M. M., Anderson, K. M., Greller, L. D., Tobin, F. L. et al. (1999). Identification and characterization of cvhsp a novel human small stress protein selectively expressed in cardiovascular and insulin-sensitive tissues. *Journal of Biological Chemistry*, 274(51):36592–36600.

Kuhn, D. E., Martin, M. M., Feldman, D. S., Terry Jr, A. V., Nuovo, G. J. e Elton, T. S. (2008). Experimental validation of mirna targets. *Methods*, 44(1):47–54.

Liang, F., Holt, I., Pertea, G., Karamycheva, S., Salzberg, S. L. e Quackenbush, J. (2000). Gene index analysis of the human genome estimates approximately 120,000 genes. *Nature genetics*, 25(2):239–240.

Liu, J., Cai, W. e Shao, X. (2011). Cancer classification based on microarray gene expression data using a principal component accumulation method. *Science China Chemistry*, 54(5):802–811.

Liu, S., Zhang, J., Xiang, Y., Zhou, W. e Xiang, D. (2019). A study of data pre-processing techniques for imbalanced biomedical data classification. *arXiv preprint arXiv:1911.00996*.

Liu, S., Zhang, J., Xiang, Y., Zhou, W. e Xiang, D. (2020). A study of data pre-processing techniques for imbalanced biomedical data classification. *International Journal of Bioinformatics Research and Applications*, 16(3):290–318.

Loomans-Kropp, H. A. e Umar, A. (2019). Cancer prevention and screening: the next step in the era of precision medicine. *NPJ precision oncology*, 3(1):1–8.

Mandal, M. e Mukhopadhyay, A. (2013). An improved minimum redundancy maximum relevance approach for feature selection in gene expression data. *Procedia Technology*, 10:20–27.

Min, S., Lee, B. e Yoon, S. (2017). Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5):851–869.

Montesinos-López, O. A., Martín-Vallejo, J., Crossa, J., Gianola, D., Hernández-Suárez, C. M., Montesinos-López, A., Juliana, P. e Singh, R. (2019). New deep learning genomic-based prediction model for multiple traits with binary, ordinal, and continuous phenotypes. *G3: Genes, genomes, genetics*, 9(5):1545–1556.

Morange, M. (2009). The central dogma of molecular biology. *Resonance*, 14(3):236–247.

Nakariyakul, S. (2016). Gene selection using interaction information for microarray-based cancer classification. Em *2016 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, páginas 1–5. IEEE.

Orozco-Arias, S., Isaza, G. e Guyot, R. (2019). Retrotransposons in plant genomes: structure, identification, and classification through bioinformatics and machine learning. *International journal of molecular sciences*, 20(15):3837.

Parker, B. J., Günter, S. e Bedo, J. (2007). Stratification bias in low signal microarray studies. *BMC bioinformatics*, 8(1):1–16.

Pauwels, E. e Vidyarthi, A. (2017). *Who will own the secrets in our genes?: A US-China race in artificial intelligence and genomics*. Woodrow Wilson International Center for Scholars Washington, DC.

Pavlidis, N. e Fizazi, K. (2009). Carcinoma of unknown primary (cup). *Critical reviews in oncology/hematology*, 69(3):271–278.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Perera, K., Chan, J. e Karunasekera, S. (2018). Feature selection for multiclass binary data. Em *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, páginas 52–63. Springer.

Popescu, M.-C., Balas, V. E., Perescu-Popescu, L. e Mastorakis, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588.

Powers, D. M. (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

Prasatha, V., Alfeilate, H. A. A., Hassanate, A., Lasassmehe, O., Tarawnehf, A. S., Alhasanatg, M. B. e Salmane, H. S. E. (2017). Effects of distance measure choice on knn classifier performance-a review. *arXiv preprint arXiv:1708.04321*, página 56.

Quackenbush, J. (2001). Computational analysis of microarray data. *Nature reviews genetics*, 2(6):418–427.

Shanmugamani, R. (2018). *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras*. Packt Publishing Ltd.

Singh, D., Febbo, P. G., Ross, K., Jackson, D. G., Manola, J., Ladd, C., Tamayo, P., Renshaw, A. A., D'Amico, A. V., Richie, J. P. et al. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer cell*, 1(2):203–209.

Singh, R., Lanchantin, J., Robins, G. e Qi, Y. (2016). Deepchrome: deep-learning for predicting gene expression from histone modifications. *Bioinformatics*, 32(17):i639–i648.

Singh, R., Lanchantin, J., Sekhon, A. e Qi, Y. (2017). Attend and predict: Understanding gene regulation by selective attention on chromatin. *Advances in neural information processing systems*, 30:6785.

Sotiriou, C., Wirapati, P., Loi, S., Harris, A., Fox, S., Smeds, J., Nordgren, H., Farmer, P., Praz, V., Haibe-Kains, B. et al. (2006). Gene expression profiling in breast cancer: understanding the molecular basis of histologic grade to improve prognosis. *Journal of the National Cancer Institute*, 98(4):262–272.

Southern, E. M. (2001). Dna microarrays. *DNA Arrays*, páginas 1–15.

Su, A. I., Welsh, J. B., Sapinoso, L. M., Kern, S. G., Dimitrov, P., Lapp, H., Schultz, P. G., Powell, S. M., Moskaluk, C. A., Frierson, H. F. et al. (2001). Molecular classification of human carcinomas by use of gene expression signatures. *Cancer research*, 61(20):7388–7393.

Suzuki, K. (2013). *Artificial neural networks: Architectures and applications*. BoD–Books on Demand.

Tabares-Soto, R., Orozco-Arias, S., Romero-Cano, V., Bucheli, V. S., Rodríguez-Sotelo, J. L. e Jiménez-Varón, C. F. (2020). A comparative study of machine learning and deep learning algorithms to classify cancer types based on microarray gene expression data. *PeerJ Computer Science*, 6:e270.

Tan, C. S., Ting, W. S., Mohamad, M. S., Chan, W. H., Deris, S. e Ali Shah, Z. (2014). A review of feature extraction software for microarray gene expression data. *BioMed research international*, 2014.

Tang, Z., Steranka, J. P., Ma, S., Grivainis, M., Rodić, N., Huang, C. R. L., Shih, I.-M., Wang, T.-L., Boeke, J. D., Fenyö, D. et al. (2017). Human transposon insertion profiling: Analysis, visualization and identification of somatic line-1 insertions in ovarian cancer. *Proceedings of the National Academy of Sciences*, 114(5):E733–E740.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tosi, S. (2009). *Matplotlib for Python developers*. Packt Publishing Ltd.

Turing, A. M. (2004). *The essential turing*. Oxford University Press.

Varoquaux, G., Raamana, P. R., Engemann, D. A., Hoyos-Idrobo, A., Schwartz, Y. e Thirion, B. (2017). Assessing and tuning brain decoders: cross-validation, caveats, and guidelines. *NeuroImage*, 145:166–179.

Von Neumann, J. (1988). John von neumann. *American Mathematical Soc*.

Walker, S. E. (2002). The human genome project. *The Canadian Journal of Hospital Pharmacy*, 55(1).

Wang, S. e Wei, J. (2017). Feature selection based on measurement of ability to classify subproblems. *Neurocomputing*, 224:155–165.

Wang, Z., Zineddin, B., Liang, J., Zeng, N., Li, Y., Du, M., Cao, J. e Liu, X. (2014). cdna microarray adaptive segmentation. *Neurocomputing*, 142:408–418.

Wong, T.-T. e Chen, D.-Q. (2011). A gene selection method for microarray data based on risk genes. *Expert Systems with Applications*, 38(11):14065–14071.

Yang, H., Pasupa, K., Leung, A. C.-S., Kwok, J. T., Chan, J. H. e King, I. (2020). *Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 18–22, 2020, Proceedings, Part V*, volume 1333. Springer Nature.

Yeoh, E.-J., Ross, M. E., Shurtleff, S. A., Williams, W. K., Patel, D., Mahfouz, R., Behm, F. G., Raimondi, S. C., Relling, M. V., Patel, A. et al. (2002). Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer cell*, 1(2):133–143.

Yue, T. e Wang, H. (2018). Deep learning for genomics: A concise overview. *arXiv preprint arXiv:1802.00810*.

Zeng, X. e Martinez, T. R. (2000). Distribution-balanced stratified cross-validation for accuracy estimation. *Journal of Experimental & Theoretical Artificial Intelligence*, 12(1):1–12.

Zohuri, B., Behgounia, F., Tanakulrungson, C., Moghaddam, M. e Ooi, E. (2021). Artificial intelligence driving heart attacks and strokes prediction. *Cardiology and Cardiovascular Research*, 2:30–41.

Zou, Q., Qu, K., Luo, Y., Yin, D., Ju, Y. e Tang, H. (2018). Predicting diabetes mellitus with machine learning techniques. *Frontiers in genetics*, 9:515.