Machine Learning: uma abordagem para classificação da expressão gênica em diferentes estágios tumorais

Giovanni Buzin Righi[1], **Sylvio André Garcia Vieira**[1]

[1]Ciência da Computação – Universidade Franciscana (UFN)
CEP 97.010-032 – Santa Maria – RS – Brazil

g.buzin@ufn.edu.br, sylvio@ufn.edu.br

Abstract. *Genetic diseases, for example breast cancer, currently affect a considerable portion of the world population. Therefore, advances in the field of molecular biology have allowed the use of methods that seek to assist in the identification of a possible predisposition to breast carcinoma, as well as the choice of more effective treatments for it. Therefore, the objective of this work is to develop a Random Forest classification model, with supervised learning, so that it is able to identify and classify samples of genes in different stages of breast cancer, correlating them with their identified groups, in order to increase the possibilities of cure with the appropriate treatment.*

Resumo. *As doenças genéticas, como por exemplo o câncer de mama, acometem atualmente uma parcela considerável da população mundial. Diante disso, avanços no campo da biologia molecular permitiram a utilização de métodos que buscam auxiliar na identificação de uma possível predisposição ao carcinoma da mama, bem como, a escolha de tratamentos mais eficazes para o mesmo. Portanto, o objetivo deste trabalho é desenvolver um modelo de classificação Random Forest, com aprendizado supervisionado, para que o mesmo seja capaz de identificar e classificar amostras de genes em diferentes estágios de câncer de mama, correlacionando-as com seus respectivos grupos, afim de aumentar as possibilidades de cura com o tratamento adequado.*
Palavras-chave*: Random Forest, mineração de dados, câncer de mama.*

1. Introdução

As doenças genéticas acometem atualmente uma parcela considerável da população mundial. Entre as mulheres, o câncer de mama é a segunda neoplasia[1] de maior incidência, ficando atrás apenas do câncer de pele não melanoma. De acordo com o Instituto Nacional do Câncer [INCA 2019], o carcinoma de mama no Brasil corresponde a cerca de 29% dos casos que atingem as mulheres. O câncer de mama também acomete homens, porém, é raro, representando apenas 1% do total de casos da doença. Um diagnóstico precoce pode contribuir para um tratamento mais eficaz. Logo, utilizar técnicas computacionais com tratamento de dados laboratoriais para detecção ou até mesmo predição de possíveis casos de câncer pode se tornar uma aliada na possibilidade de antecipar um tratamento bem como prover uma melhoria na qualidade de vida do paciente.

[1]É uma proliferação anormal, autônoma e descontrolada de um determinado tecido do corpo, mais conhecida como tumor.

No ramo das ciências biológicas, a aplicação bem sucedida da mineração de dados fornece novos conhecimentos biomédicos que podem ser usados efetivamente para apoiar a tomada de decisões clinicas, como por exemplo, o processo de diagnóstico, na escolha de opções de tratamento e também para a previsão de prognóstico [Yoo et al. 2012]. Aplicações clinicas em mineração de dados possui algumas singularidades: (i) as decisões devem ser cuidadosamente avaliadas; (ii) os experimentos geralmente são muito caros; (iii) os dados podem ser afetados por várias fontes de incerteza, incluindo erros nos dados de entrada bem como dados incompletos. Por essa razão, tem-se modelos individuais para cada paciente, que podem ajudar a fornecer uma interpretação correta dos eventos em análise [Bellazzi et al. 2011].

O aprendizado de máquina é um método que se concentra em como os computadores aprendem com os dados. Os algoritmos analisam informações de entrada, as processam e preveem possíveis saídas, à medida que, tentam diferentes abordagens e otimizam sua capacidade de chegar ao resultado mais preciso. Na análise dos dados para classificação, o algoritmo pode trabalhar de forma supervisionada, ou seja, com um atributo alvo que orienta toda a classificação [Pang-Ning et al. 2009]. Devido a acurácia de seus algoritmos para classificar padrões, pesquisadores passaram a aplicá-los na resolução de muitos problemas clínicos, como o diagnóstico de uma patologia e a previsão de prognóstico para um paciente, que dependem de uma interação complexa de muitas variáveis [Deo 2015].

1.1. Objetivo geral

O objetivo deste trabalho é desenvolver um modelo de classificação *Random Forest* capaz de identificar e classificar amostras de genes em diferentes estágios de câncer de mama correlacionando-as aos seus respectivos grupos, permitindo uma detecção precoce da doença, aumentando as possibilidades de cura.

1.2. Objetivos Específicos

1. Estudar linguagem de programação Python bem como suas bibliotecas para área da bioinformática;
2. Estudar linguagem de programação R e seus pacotes para construção de modelos de classificação;
3. Estudar algoritmos de aprendizado de máquina;
4. Estudar e aplicar técnicas de mineração de dados;
5. Aplicar as bibliotecas Pandas, do Python e randomForest, do R;
6. Desenvolver um classificador *Random Forest* com aprendizado supervisionado;
7. Analisar amostras de expressão gênica, identificando o quão estas estão diferencialmente expressas com relação as amostras controle.
8. Avaliar os resultados.

2. Referencial Teórico

Nesta seção serão apresentados os conceitos e métodos utilizados para o desenvolvimento deste trabalho, bem como assuntos relacionados com o tema proposto que buscam auxiliar para o entendimento do mesmo.

2.1. Genes

Os genes são segmentos de moléculas de DNA responsáveis pelas características herdadas geneticamente. Estão organizados em estruturas chamadas de cromossomos, que por sua vez, contém longas cadeias de DNA, onde se distribuem os genes. Os seres humanos têm aproximadamente 20.000 genes [Institute 2020].

Dentro da genética moderna, o gene é uma fração de nucleotídeos do DNA que pode ser transcrita para RNA. O Ácido ribonucleico é uma molécula formada por nucleotídeos que intervem em diferentes funções biológicas, como a síntese de proteínas, onde um erro ou dano causado à um gene pode vir a provocar uma doença genética [Pearson 2006]. As mutações podem ocorrer quando um nucleotídeo é excluído, replicado incorretamente, ou ainda substituído, à medida que para cada alteração têm-se um efeito próprio. A mutação geralmente tem pouco ou nenhum efeito, mas, quando altera um organismo, a mudança pode ser letal ou causar doenças [Britannica 2020].

2.1.1. Expressão Gênica

A expressão gênica é o processo no qual as características hereditárias de um gene passam a serem utilizadas para auxiliar na montagem de uma molécula de proteína. A célula analisa a sequência do gene por meio de grupos de três bases, chamados de códon, onde cada conjunto corresponde a um diferente aminoácidos que pode vir a ser usado para a construção de proteínas [Institute 2020].

O processo de codificação de um polipeptídeo por parte de um gene é expresso em duas etapas [Academy 2020], são elas:

- Transcrição: uma fita do DNA do gene é copiada para RNA, o qual, deve passar por etapas adicionais de processamento para se tornar um RNA mensageiro;
- Tradução: a sequência de nucleotídeos do RNA mensageiro é decodificada para especificar a sequência de aminoácidos de um polipeptídeo.

2.2. Câncer de Mama

O câncer pode ser definido como sendo um crescimento desordenado de células capazes de invadir tecidos e órgãos vizinhos. Dividindo-se rapidamente, estas células tendem a ser muito agressivas e incontroláveis. Conhecer informações a respeito dos diferentes perfis da doença pode auxiliar para um planejamento eficiente na elaboração de programas para prevenção e controle da mesma [INCA 2019].

O câncer de mama pode ser detectado em fases iniciais, em grande parte dos casos, aumentando assim a possibilidade de tratamentos menos agressivos e com taxas de sucesso satisfatórias. Um nódulo[2] ou outro sintoma suspeito deve ser investigado para confirmar se é ou não câncer de mama [INCA 2019].

Com os avanços no campo da biologia molecular, atualmente existem diversos tratamentos para o câncer, como por exemplo, a quimioterapia, radioterapia e a imunoterapia. Por conseguinte, existem diferentes formas de tumores [Lodge 2020], das quais podemos destacar:

[2]Crescimentos anormais na pele ou qualquer tecido do corpo, formando elevações.

- Tumor quente: apresentam sinais de inflamação, significando que o tumor já foi infiltrado por células T[3] que passam a combater a neoplasia. Normalmente respondem bem ao tratamento imunoterápico;
- Tumor intermediário: relacionado ao estadiamento do tumor, se tem ou não metástase. Isto diz respeito ao tipo de tratamento que deve ser feito;
- Tumor frio: as células cancerígenas não foram infiltradas com células T, um sinal de que a resposta imune não esta funcionando. A ausência de células T torna difícil um resultado positivo com a imunoterapia, levando à utilização de tratamentos mais tradicionais como a quimioterapia.

2.3. Imunoterapia

A imunoterapia é um tipo de tratamento que consiste em ajudar o sistema imunológico a identificar e combater células cancerígenas. O tratamento consiste em isolar linfócitos que infiltraram o tumor e selecionar aqueles capazes de identificar as proteínas produzidas somente pelas células tumorais. Muitos tratamentos imunoterápicos para prevenir ou tratar diferentes tipos de câncer também podem ser usados em combinação com cirurgias, quimioterapias e radioterapias direcionadas para melhorar sua eficácia [NIH 2020].

A imunoterapia nem sempre funciona para todos os pacientes, pesquisadores estão desenvolvendo maneiras de determinar quais pacientes provavelmente responderão ao tratamento e quais não, levando a novas estratégias para expandir o número de pacientes que podem vir a se beneficiar potencialmente do tratamento imunoterápico [Research 2020].

2.4. Bancos de Dados Biológicos

O crescimento da produção de dados biológicos em grande escala concomitante ao aumento exponencial da literatura biomédica disponível trouxeram a necessidade de um meio comum capaz de gerenciar e processar estas informações [Lu 2011].

Deste modo, o *National Center for Biotechnology Information* (NCBI) foi encarregado de criar sistemas automatizados para armazenar e analisar conhecimentos sobre biologia molecular, bioquímica e genética. Dentre os bancos de dados biológicos desenvolvidos, o *Gene Expression Omnibus* (GEO), é um repositório público que armazena conjuntos de dados genômicos e disponibiliza ferramentas para que seus usuários sejam capazes de consultar e baixar as informações requeridas [Clough and Barrett 2016].

2.5. Random Forest

Random Forest é um algoritmo de aprendizagem supervisionada que pode ser aplicado tanto para métodos de classificação como para regressão. Sua abordagem, cria várias árvores de decisão aleatória e as combina para obter uma predição mais estável e com maior acurácia [Scornet et al. 2015].

Devido a sua precisão para classificar e identificar padrões, bem como, a possibilidade de utilizá-lo em diversos problemas de previsão, pesquisadores passaram a aplicar o algoritmo na área da bioinformática para a resolução de problemas clínicos, como por exemplo, o diagnóstico de uma patologia e a previsão de prognóstico para um paciente [Biau and Scornet 2016].

[3]Células do sistema imunológico, grupo de glóbulos brancos (leucócitos) responsáveis pela defesa do organismo contra agentes desconhecidos

2.6. Machine Learning

O aprendizado de máquina é uma aplicação de inteligência artificial que se caracteriza pelo desenvolvimento de sistemas com capacidade de aprender e melhorar automaticamente, sem a necessidade de ser programado explicitamente. Para tanto, o processo de aprendizado pode ter como base a observação de padrões em conjuntos de dados buscando uma melhora quanto a tomada de decisões sobre um determinado assunto. Como exemplo, destaca-se os computadores utilizados para detectar e prever casos de neoplasias a partir de relatórios de investigação médica de um paciente [Ray 2019].

Dentre as técnicas para o aprendizado de máquina, há o aprendizado não supervisionado, onde apenas a entrada de dados é fornecida no procedimento de treinamento, e o aprendizado supervisionado, no qual utiliza-se a entrada de dados com suas respectivas saídas, as quais devem estar corretas. Esse método se baseia na minimização do erro entre o valor desejado de saída e o valor computado [Lima et al. 2016].

2.7. Mineração de Dados

A mineração de dados é uma das etapas da Descoberta de Conhecimento em Bases de Dados (*Knowledge Discovery in Databases - KDD*), onde é realizada a busca efetiva por conhecimentos úteis [Goldschmidt and Passos 2005].

Segundo [Pang-Ning et al. 2009], as técnicas de mineração de dados podem ser organizadas para agir sobre grandes bancos de dados com o intuito de descobrir padrões que poderiam, de outra forma, permanecer ignorados. Algoritmos genéticos e modelos estatísticos e probabilísticos, são exemplos de técnicas que podem ser utilizadas na etapa de mineração.

2.8. Linguagem de Programação

Para desenvolver este trabalho foram utilizados Python, visto que é uma linguagem de programação de grande desempenho [Python 2019] e R, por ser uma linguagem voltada à manipulação, análise e visualização de dados [R 2020]. Bem como as seguintes bibliotecas:

- Pandas: biblioteca em código aberto que fornece estruturas de dados de alto desempenho e ferramentas de análise de dados para a linguagem de programação Python [Pandas 2019];
- RandomForest: pacote da linguagem R que possibilita a implementação de métodos tanto para classificação como para regressão [RandomForest 2020].

3. Trabalhos Correlatos

Nesta seção serão apresentados os artigos que auxiliaram para o desenvolvimento deste trabalho.

3.1. Classifying Osteosarcoma Using Meta-Analysis of Gene Expression

No trabalho de [Alge et al. 2018], utilizou-se como base de dados o repositório do *Gene Expression Omnibus* (GEO) juntamente com as amostras coletadas de pacientes com osteossarcoma da Universidade *Jiao Tong* (SJTU), de Xangai.

Isto posto, elaborou-se uma lista com 209 genes de interesse associados ao osteossarcoma, filtrou-se a mesma, obtendo assim uma nova relação contendo 178 genes. Dividiu-se aleatoriamente este conjunto de dados reduzido em treinamento e teste. Sucessivamente, desenvolveu-se um classificador *Random Forest* com validação cruzada de 3 vezes para estimar o desempenho do modelo de classificação, tal analise foi realizada usando a linguagem de programação R. Por conseguinte, alcançou-se como resultado para o conjunto de genes aplicados no treinamento uma precisão média de 74,1%, enquanto para o conjunto utilizado para teste constatou-se uma precisão de 80,0%.

3.2. A Random Forest classifier-based approach in the detection of abnormalities in the retina

Em [Chowdhury et al. 2019], comparou-se o desempenho de um classificador *Random Forest* em relação a um classificador *Naive Bayes*[4], buscando o modelo mais preciso capaz de auxiliar à médicos oftalmologistas e pesquisadores da área na detecção de anormalidades na retina.

Para tal, criou-se um banco de dados composto com imagens referentes a degeneração macular e a retinopatia diabética, o acervo possuía cerca de 405 imagens. Assim sendo, dividiu-se os conjuntos de dados à serem utilizados para o treinamento e teste dos classificadores. Em suma, como resultados obtidos apurou-se uma maior efetividade do classificador *Random Forest*, obtendo 93,5% de precisão, enquanto o classificador *Naive Bayes* atingiu uma precisão de 83,6%.

3.3. Random Forests for Diabetes Diagnosis

No estudo feito por [Benbelkacem and Atmani 2019], analisou-se a precisão de um modelo de classificação *Random Forest* em comparação com algoritmos para aprendizagem de máquinas, destaca-se o uso dos algoritmos C4.5[5] e máquinas de vetores de suporte[6] (SVM). A abordagem proposta concentrou-se em fornecer um diagnóstico adequado para auxiliar os profissionais que atuam no tratamento da diabetes.

O conjunto de dados utilizado fundamentou-se nas informações retiradas da *UCI Machine Learning Repository* e possuía amostras de 768 pacientes mulheres, à medida que, 268 delas apresentavam diabetes. Dividiu-se a base de dados em treinamento e teste, bem como, aplicou-se validação cruzada de 5 vezes no modelo de classificação afim de estimar seu desempenho. Como resultado, constatou-se uma melhor performance com o classificador *Random Forest*, no qual, alcançou-se uma taxa de erro de 0.21 perante a identificação de pacientes diabéticos. Todavia, para os algoritmos C4.5 e SVM, atingiu-se uma taxa de erro de 0.23 e 0.25, respectivamente.

Em virtude dos resultados apresentados na subseção 3.1, onde utilizou-se um modelo baseado em *Random Forest*, e subseções 3.2 e 3.3 em que comparou-se o desempenho de classificadores *Random Forest* com algoritmos de aprendizagem de máquinas, conclui-se que tais funcionalidades e técnicas são capazes de auxiliar este trabalho, uma

[4]Classificador probabilístico, aplicado em processamento de linguagem natural e diagnósticos médicos

[5]Algoritmo utilizado para criar uma modelo de árvore de decisão, utilizados como classificadores estatísticos

[6]Método de classificação baseado em um algoritmo de otimização que define hiperplanos de separação ótimos entre amostras.

vez que alcançaram evidências positivas de que podem ser aplicadas de maneira efetiva como ferramentas de classificação.

4. Metodologia

Este trabalho foi desenvolvido por meio de um processo dividido em duas partes, a primeira foi implementada utilizando a linguagem de programação Python junto com a biblioteca pandas. Nesta etapa, foi feita a extração dos dados e o pré-processamento destas informações, preparando-as para serem usadas na segunda parte do projeto.

Na segunda parte, com os dados resultantes gerados na primeira etapa, foi desenvolvido um modelo de classificação *Random Forest*, bem como, fez-se uso de técnicas de mineração de dados afim de avaliar o desempenho do modelo classificador. Para tal utilizou-se a linguagem de programação R junto com o pacote randomForest.

A base de dados utilizada neste trabalho foi extraída do banco de dados biológico *Gene Expression Omnibus* (GEO), disponibilizada junto ao artigo de [Tekpli et al. 2019][7], cujo propósito era analisar o ambiente imunológico em torno das células cancerígenas do carcinoma de mama para inibir o crescimento tumoral e dessa forma, ajudar a identificar pacientes que poderiam se beneficiar perante tratamentos imunoterápicos. Para tal, dividiu-se as amostras de genes utilizados em três grupos: (i) cluster A, relativo aos tumores frios; (ii) cluster B, referente aos tumores intermediários; e por fim (iii) cluster C, alusivo aos tumores quentes.

Em suma, na Figura 1, é apresentada a função desenvolvida para o refinamento dos dados. Utilizando um filtro, como sendo uma lista contendo as informação que devem ser retiradas do arquivo (valores nulos e com notação científica), percorre-se o mesmo por meio de um laço de repetição, verificando se algum conteúdo do filtro está presente no arquivo, caso exista, é removido, até que não se localize mais nenhum. Por fim o novo conteúdo é salvo em um novo arquivo no formato *'csv'*.

```
# Método para excluir linhas que apresentam 0 e E- por meio de um filtro.
def data_clean():
    find_values = ['0;0;', 'E-', ';0;']
    with open('data/data.csv') as data, open('data/dataClean.csv', 'w') as newData:
        for line in data:
            if not any(find_values in line for find_values in find_values):
                newData.write(line)
```

Figura 1. Refinando atributos do arquivo.

Após a limpeza dos dados, foi utilizado o *Numbers*[8] para dividir o arquivo de acordo com os grupos citados anteriormente, e assim calcular a média para cada valor dos genes.

A Figura 2, mostra a função desenvolvida para calcular a média dos valores dos gene de cada grupo. Para isso leram-se os arquivos *'csv'* e por meio da biblioteca pandas do Python e um de seus métodos, *'mean'*, foi calculada a média, afim de tornar possível o cálculo da diferença de expressão gênica.

[7] https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE135298

[8] Aplicativo da *Apple* para criação de planilhas.

```python
# Método para calcular a média de cada gene presente no ClusterA, ClusterB e ClusterC.
def data_average():
    cluster_a_data = pd.read_csv("dataModified/dataAMod.csv", delimiter=';', index_col=0, low_memory=False)
    cluster_a_transpose = cluster_a_data.transpose()
    # print(cluster_a_transpose)

    cluster_b_data = pd.read_csv("dataModified/dataBMod.csv", delimiter=';', index_col=0, low_memory=False)
    cluster_b_transpose = cluster_b_data.transpose()
    # print(cluster_b_transpose)

    cluster_c_data = pd.read_csv("dataModified/dataCMod.csv", delimiter=';', index_col=0, low_memory=False)
    cluster_c_transpose = cluster_c_data.transpose()
    # print(cluster_c_transpose)

    cluster_a_average = cluster_a_transpose.mean()
    # print(cluster_a_average)
    cluster_a_average.to_csv('data/dataAverageA.csv', header=False)

    cluster_b_average = cluster_b_transpose.mean()
    # print(cluster_b_average)
    cluster_b_average.to_csv('data/dataAverageB.csv', header=False)

    cluster_c_average = cluster_c_transpose.mean()
    # print(cluster_c_average)
    cluster_c_average.to_csv('data/dataAverageC.csv', header=False)
```

Figura 2. Cálculo da média de cada gene [Autor].

A Figura 3, após o cálculo da média dos valores dos genes de cada grupo, apresenta a função que efetua a diferença dos valores das médias e além disso, seleciona as 300 maiores diferenças gênicas encontradas. Adotando o cluster C, relativo aos tumores quentes, como modelo de referência, lê-se novamente os arquivos *'csv'* de cada grupo, e subtrai-se os valores do cluster C com os valores do cluster A e cluster B, após, fazendo uso do método *'nlargest'* do pandas seleciona-se as 300 maiores diferenças encontradas, com o intuito de preparar os dados para serem comparados.

```python
# Método para calcular a diferença entre os valores da média encontrados referente ao Cluster C com os demais Clusters.
# Rankear as 300 maiores expressões gênicas encontradas.
def data_difference_ranking():
    cluster_a_difference = pd.read_csv("dataModified/dataAverageAMod.csv", delimiter=';', index_col=0, low_memory=False)
    difference_a_transpose = cluster_a_difference.transpose()
    # print(difference_a_transpose)

    cluster_b_difference = pd.read_csv("dataModified/dataAverageBMod.csv", delimiter=';', index_col=0, low_memory=False)
    difference_b_transpose = cluster_b_difference.transpose()
    # print(difference_b_transpose)

    cluster_c_difference = pd.read_csv("dataModified/dataAverageCMod.csv", delimiter=';', index_col=0, low_memory=False)
    difference_c_transpose = cluster_c_difference.transpose()
    # print(difference_c_transpose)

    difference_ca = difference_c_transpose - difference_a_transpose.values
    # print(difference_ca)
    difference_ca_transpose = difference_ca.transpose()
    difference_ca_transpose.to_csv('data/dataDifferenceCA.csv', header=True)

    difference_ca_ranked = difference_ca_transpose.nlargest(300, ['CLUSTER C'])
    # print(difference_ca_ranked)
    difference_ca_ranked.to_csv('data/dataRankedDifferenceCA.csv', header=True)

    difference_cb = difference_c_transpose - difference_b_transpose.values
    # print(difference_cb)
    difference_cb_transpose = difference_cb.transpose()
    difference_cb_transpose.to_csv('data/dataDifferenceCB.csv', header=True)

    difference_cb_ranked = difference_cb_transpose.nlargest(300, ['CLUSTER C'])
    # print (difference_cb_ranked)
    difference_cb_ranked.to_csv('data/dataRankedDifferenceCB.csv', header=True)
```

Figura 3. Cálculo da diferença entre os valores da média [Autor].

Na Figura 4, mostra a função que faz a comparação dos genes em comuns presentes nos arquivos gerados da diferença dos valores entre o cluster C com os demais grupos e, após, seleciona-se os 100 primeiros genes encontrados. Para isso os arquivos são lidos e

utilizam-se os métodos do pandas *'isin'* para fazer a comparação e *'head'* para selecionar os 100 primeiros genes.

```python
# Método que verifica os genes presentes nos dois arquivos .csv da diferença do Cluster C com os demais Clusters.
# Selecionar os 100 primeiros genes.
def data_difference_compare():
    difference_ca_ranked = pd.read_csv("dataModified/dataRankedDifferenceCAMod.csv", delimiter=';', index_col=0,
                                       low_memory=False)
    ranked_ca_compare = difference_ca_ranked.drop(columns="DifferenceCA")
    # print(difference_ca_ranked.columns)
    # print(difference_ca_ranked)

    difference_cb_ranked = pd.read_csv("dataModified/dataRankedDifferenceCBMod.csv", delimiter=';', index_col=0,
                                       low_memory=False)
    ranked_cb_compare = difference_cb_ranked.drop(columns="DifferenceCB")
    # print(difference_cb_ranked.columns)
    # print(difference_cb_ranked)

    difference_compare = (ranked_ca_compare[ranked_ca_compare.Genes.isin(ranked_cb_compare.Genes)]).head(100)
    # print(difference_compare)
    difference_compare.to_csv('data/dataResultDifferenceCompare.csv', header=True)
```

Figura 4. Comparação dos genes comuns nos arquivos da diferença dos valores das médias [Autor].

A Figura 5, apresenta a função que compara o resultado dos genes em comuns encontrados anteriormente com os arquivos *'csv'* de cada cluster e novamente seleciona-se os 100 primeiros genes. Para isso, lê-se os dados de cada grupo e mais uma vez utilizam-se os métodos do pandas *'isin'* para fazer a comparação e *'head'* para selecionar os 100 primeiros genes.

```python
# Método que compara os genes em comum na diferença entre Cluster CA e Cluster CB com os genes dos Clusters A,B e C.
# Selecionar os 100 primeiros genes.
def data_cluster_compare():
    cluster_a_data = pd.read_csv("dataModified/dataClusterAMod.csv", delimiter=';', index_col=0, low_memory=False)
    # print(cluster_a_data)

    cluster_b_data = pd.read_csv("dataModified/dataClusterBMod.csv", delimiter=';', index_col=0, low_memory=False)
    # print(cluster_b_data)

    cluster_c_data = pd.read_csv("dataModified/dataClusterCMod.csv", delimiter=';', index_col=0, low_memory=False)
    # print(cluster_c_data)

    difference_compare = pd.read_csv("dataModified/dataResultDifferenceCompareMod.csv", delimiter=';', index_col=0,
                                     low_memory=False)
    # print(difference_compare)

    compare_a_result = (cluster_a_data[cluster_a_data.Genes.isin(difference_compare.Genes)]).head(100)
    # print(compare_a_result)
    compare_a_result.to_csv('data/dataCompareClusterA.csv', header=True, index=False)

    compare_b_result = (cluster_b_data[cluster_b_data.Genes.isin(difference_compare.Genes)]).head(100)
    # print(compare_b_result)
    compare_b_result.to_csv('data/dataCompareClusterB.csv', header=True, index=False)

    compare_c_result = (cluster_c_data[cluster_c_data.Genes.isin(difference_compare.Genes)]).head(100)
    # print(compare_c_result)
    compare_c_result.to_csv('data/dataCompareClusterC.csv', header=True, index=False)
```

Figura 5. Comparação do resultado dos genes em comuns encontrados com cada cluster [Autor].

A Figura 6, mostra a função que prepara os dados para a etapa de mineração. Lê-se o arquivo *'csv'* referente a comparação dos genes em comuns presentes nos arquivos gerados da diferença dos valores entre o cluster C com os demais grupos, selecionando agora os 14 primeiros genes, e aplica-se o resultado nos arquivos dos cluster A, B e C.

Utilizando os métodos da biblioteca pandas, a seleção dos 14 genes é feita com *'head'* e a comparação com *'isin'*, armazenando o resultado em um novo arquivo à ser aplicado na segunda parte deste trabalho.

```
# Método que faz o rank dos primeiros 14 genes presentes nos arquivos .csv referentes aos Clusters A, B e C.
def cluster_data_mining():
    difference_compare = pd.read_csv("dataModified/dataResultDifferenceCompareMod.csv", delimiter=';', index_col=0,
                                     low_memory=False).head(14)
    # print(difference_compare)

    cluster_a_data = pd.read_csv("dataModified/dataClusterAMod.csv", delimiter=';', index_col=0, low_memory=False)
    # print(cluster_a_data)

    cluster_b_data = pd.read_csv("dataModified/dataClusterBMod.csv", delimiter=';', index_col=0, low_memory=False)
    # print(cluster_b_data)

    cluster_c_data = pd.read_csv("dataModified/dataClusterCMod.csv", delimiter=';', index_col=0, low_memory=False)
    # print(cluster_c_data)

    mining_a_compare_result = (cluster_a_data[cluster_a_data.Genes.isin(difference_compare.Genes)])
    # print(mining_a_compare_result)
    mining_a_compare_result.to_csv('dataMining/dataMiningCompareA.csv', header=True)

    mining_b_compare_result = (cluster_b_data[cluster_b_data.Genes.isin(difference_compare.Genes)])
    # print(mining_b_compare_result)
    mining_b_compare_result.to_csv('dataMining/dataMiningCompareB.csv', header=True)

    mining_c_compare_result = (cluster_c_data[cluster_c_data.Genes.isin(difference_compare.Genes)])
    # print(mining_c_compare_result)
    mining_c_compare_result.to_csv('dataMining/dataMiningCompareC.csv', header=True)
```

Figura 6. Comparação dos genes em comuns e preparação dos dados para a mineração de dados [Autor].

Na Figura 7, utilizam-se os dados gerados na primeira parte deste trabalho, porém, agora na segunda etapa, passa a ser utilizada a linguagem de programação R. Primeiramente, instala-se o pacote *'randomForest'* e executa-se o comando *'library'* para começar a usar a biblioteca, após, seleciona-se o diretório a ser trabalhado com o comando *'setwd'*.

O conteúdo do arquivo *'csv'* gerado foi divido em treinamento e teste, além disso, a coluna relativa ao atributo 'Amostras' foi removida por meio de um *'select'*, devido ao fato da mesma conter valores utilizados como identificadores das amostras, a sua presença poderia atrapalhar o modelo no momento da classificação. Utilizando o pacote *'randomForest'* criou-se o modelo classificador, definindo-se os parâmetros: (i) fórmula, no qual se determina o atributo da classe e quais são os atributos descritivos do conjunto de dados; (ii) data, que indica qual dados serão usados; e (iii) ntree, referente ao número de árvores à serem aplicadas no modelo.

Em síntese, utilizou-se a função *predict*, afim de classificar o modelo *Random Forest*, definindo-se os parâmetros: (i) modelo, referente ao resultado da construção do modelo de classificação *Random Forest*; (ii) exemplares, no qual contém os atributos descritivos do arquivo *'csv'* de teste; e (iii) classe, que indica como será apresentado o resultado da predição do valores. Por último, executa-se o comando *'table'* para mostrar os resultados da predição em formato tabelado, facilitando sua análise. O processo é replicado para todos os clusters.

```
install.packages("randomForest")
library("randomForest")

setwd("~/GiovanniBuzinTFGR")

read_cluster_train <- read.csv("dataMiningClusterTreino.csv", head=TRUE, sep=";")
read_clusterA_test <- read.csv("clusterATeste.csv", head=TRUE, sep=";")
#read_clusterA_test <- read.csv("clusterATesteSix.csv", head=TRUE, sep=";")

train <- subset(read_cluster_train,select=c(-Amostras))
test <- subset(read_clusterA_test,select=c(-Amostras))

train_cluster_data <- train[1:1330,]
test_clusterA_data <- test[1:14,]
#test_clusterA_data <- test[1:84,]

random_forest_cluster <- randomForest(Grupo~Genes+Valor, data=train_cluster_data, ntree=1000)
random_forest_cluster

#random_forest_plot <- plot(random_forest_cluster, type="l", main=deparse(substitute(random_forest_cluster)))
#random_forest_plot

#supplied_test_clusterA <- predict(random_forest_cluster, test_clusterA_data, "prob")

supplied_test_clusterA <- predict(random_forest_cluster, test_clusterA_data, "class")
supplied_test_clusterA

table(supplied_test_clusterA, test_clusterA_data$Grupo)
```

Figura 7. Aplicação do modelo de classificação *Random Forest* e técnicas para a mineração de dados [Autor].

5. Resultados

Nesta seção, são apresentados os resultados que foram obtidos durante o desenvolvimento deste trabalho. A Figura 8 apresenta os resultados obtidos por meio do classificador *Random Forest*, em que são expressos: (i) o tipo e o número de árvores utilizadas no modelo; (ii) a taxa de erro que o modelo obteve perante o processo de classificação; e (iii) a matriz de confusão que foi gerada a partir dos resultados encontrados.

Na matriz de confusão[9], utilizando o arquivo *'csv'* de treinamento, o qual possuía 1330 amostas de genes, e com o atributo da classe definido como sendo a coluna 'Grupo', para a primeira linha da matriz, o classificador previu corretamente 712 amostras como sendo realmente do cluster A, contudo, classificou incorretamente 28 amostras como sendo do cluster B e também de maneira incorreta classificou 44 amostras como sendo do cluster C. Para a segunda linha da matriz, o classificador previu corretamente 61 amostras como sendo realmente do cluster B, porém, classificou de maneira errônea 175 amostras como sendo do cluster A e do mesmo modo, classificou de forma incorreta 44 amostras como sendo do cluster C. Por fim, para a última linha da matriz, o classificador previu corretamente 147 amostras como sendo realmente do cluster C, assim como, classificou incorretamente 80 amostras como sendo do cluster A e também de maneira incorreta classificou 39 amostras como sendo do cluster B. Este processo de classificação foi obtido com uma taxa de erro de 30,83%.

As Figura 9, 10 e 11, fazem referência a análise do desempenho do classificador *Random Forest*. Este trabalho, utilizando o arquivo *'csv'* de teste, o qual possuía 14 amostras selecionadas do arquivo de treinamento, analisou a capacidade do modelo de classificação *Random Forest* para correlacionar estes dados à seus respectivos grupos. O

[9]Na matriz de confusão, lê-se na diagonal principal as classificações corretas e as demais posições da matriz representa as classificações erradas

```
Call:
 randomForest(formula = Grupo ~ Genes + Valor, data = train_cluster_data,      ntree = 1000)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 1

        OOB estimate of  error rate: 30.83%
Confusion matrix:
    A  B   C class.error
A 712 28  44  0.09183673
B 175 61  44  0.78214286
C  80 39 147  0.44736842
```

Figura 8. Resultado referente ao classificador *Random Forest*.

processo foi repetido em todos os clusters.

Para a Figura 9, o classificador identificou corretamente as 14 amostras como sendo realmente do cluster A. Para a Figura 10, o classificador identificou corretamente 4 amostras como sendo realmente do cluster B, as outras 10 amostras foram classificadas incorretamente como sendo do cluster A. Por último, para a Figura 11, o classificador identificou corretamente 13 amostras como sendo realmente do cluster C, à medida que, 1 amostra foi incorretamente associada ao cluster A.

```
> supplied_test_clusterA <- predict(random_forest_cluster, test_clusterA_data, "class")
> supplied_test_clusterA
 1  2  3  4  5  6  7  8  9 10 11 12 13 14
 A  A  A  A  A  A  A  A  A  A  A  A  A  A
Levels: A B C
> table(supplied_test_clusterA, test_clusterA_data$Grupo)

supplied_test_clusterA  A
                     A 14
                     B  0
                     C  0
```

Figura 9. Avaliação dos resultados de classificação referente ao cluster A.

```
> supplied_test_clusterB <- predict(random_forest_cluster, test_clusterB_data, "class")
> supplied_test_clusterB
 1  2  3  4  5  6  7  8  9 10 11 12 13 14
 B  A  A  B  B  A  A  B  A  A  A  A  A  A
Levels: A B C
> table(supplied_test_clusterB, test_clusterB_data$Grupo)

supplied_test_clusterB  B
                     A 10
                     B  4
                     C  0
```

Figura 10. Avaliação dos resultados de classificação referente ao cluster B.

```
> supplied_test_clusterC <- predict(random_forest_cluster, test_clusterC_data, "class")
> supplied_test_clusterC
 1  2  3  4  5  6  7  8  9 10 11 12 13 14
 C  C  C  C  C  C  C  C  A  C  C  C  C  C
Levels: A B C
> table(supplied_test_clusterC, test_clusterC_data$Grupo)

supplied_test_clusterC  C
                     A  1
                     B  0
                     C 13
```

Figura 11. Avaliação dos resultados de classificação referente ao cluster C.

6. Conclusão e Trabalhos Futuros

No decorrer deste trabalho, houve uma dedicação em formar um sólido embasamento teórico sobre o tema abordado. Por conseguinte, foi possível obter os conhecimentos referentes as ferramentas e tecnologias a serem utilizadas. Também foi possível obter o conhecimento necessário para a elaboração do referencial teórico, em que buscou-se explicar os conceitos e métodos posteriormente aplicados no desenvolvimento do presente trabalho, bem como para introduzir as técnicas utilizadas pelos autores correlatos que demonstraram eficiência quando empregadas como ferramentas de classificação.

Este trabalho tomou como base diferentes amostras de expressões gênicas, com as quais, com o apoio da linguagem de programação Python, bem como a biblioteca pandas foi possível a análise destes dados e a identificação das amostras mais diferencialmente expressas. Foi desenvolvido, aplicando a linguagem de programação R e a biblioteca randomForest, um modelo de classificação *Random Forest*, com aprendizado supervisionado, em que foi possível classificar e identificar os diferentes estágios tumorais em torno das amostras dos genes, com uma taxa de acerto de aproximadamente 70%. Com os resultados alcançados pelo modelo, utilizando técnicas de mineração de dados, foi possível verificar o desempenho do mesmo, demonstrando que o modelo de classificação *Random Forest* pode vir a auxiliar na identificação precoce de pacientes que poderiam vir a se beneficiar perante tratamentos imunoterápicos.

Para os trabalhos futuros é possível a implementação de um classificador aplicando os métodos apresentados neste trabalho, afim de validar o mesmo em diferentes bases de dados. Concomitante a isto, pode ser utilizado algum banco de dados para importar e manipular as informações requeridas, à medida que, poderia ajudar a reduzir o número de arquivos *'csv'* utilizados neste trabalho.

Referências

Academy, K. (2020). Introdução à expressão dos genes (dogma central). https://pt.khanacademy.org/science/biology/gene-expression-central-dogma/central-dogma-transcription/a/intro-to-gene-expression-central-dogma/

Alge, O., Gryak, J., Hua, Y., and Najaria, K. (2018). Classifying osteosarcoma using meta-analysis of gene expression. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2400–2404. IEEE.

Bellazzi, R., Ferrazzi, F., and Sacchi, L. (2011). Predictive data mining in clinical medicine: a focus on selected methods and applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(5):416–430.

Benbelkacem, S. and Atmani, B. (2019). Random forests for diabetes diagnosis. In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–4. IEEE.

Biau, G. and Scornet, E. (2016). A random forest guided tour. *Test*, 25(2):197–227.

Britannica, E. (2020). Gene heredity. https://www.britannica.com/science/gene/.

Chowdhury, A. R., Chatterjee, T., and Banerjee, S. (2019). A random forest classifier-based approach in the detection of abnormalities in the retina. *Medical & biological engineering & computing*, 57(1):193–203.

Clough, E. and Barrett, T. (2016). The gene expression omnibus database. In *Statistical Genomics*, pages 93–110. Springer.

Deo, R. C. (2015). Machine learning in medicine. *Circulation*, 132(20):1920–1930.

Goldschmidt, R. and Passos, E. (2005). *Data mining: um guia prático*. Gulf Professional Publishing.

INCA (2019). Câncer de mama. https://www.inca.gov.br/tipos-de-cancer/cancer-de-mama.

Institute, N. H. G. R. (2020). Gene. https://www.genome.gov/genetics-glossary/Gene/.

Lima, I., Pinheiro, C. A., and Santos, F. A. O. (2016). *Inteligência artificial*, volume 1. Elsevier Brasil.

Lodge, A. (2020). Immunology for non-immunologists: Hot vs. cold tumors. https://cellero.com/blog/immunology-for-non-immunologists-hot-vs-cold-tumors/.

Lu, Z. (2011). Pubmed and beyond: a survey of web tools for searching biomedical literature. *Database*, 2011.

NIH (2020). Immunotherapy to treat cancer. https://www.cancer.gov/about-cancer/treatment/types/immunotherapy/.

Pandas (2019). Pandas. https://pandas.pydata.org/.

Pang-Ning, T., Steinbach, M., and Kumar, V. (2009). Introdução ao "data mining"-mineração de dados. *Rio de Janeiro: Editora Ciência Moderna*.

Pearson, H. (2006). What is a gene?

Python (2019). What is python? executive summary. https://www.python.org/doc/essays/blurb/.

R (2020). R language definition. https://cran.r-project.org/doc/manuals/r-release/R-lang.html/.

RandomForest (2020). Randomforest. https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest/.

Ray, S. (2019). A quick review of machine learning algorithms. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pages 35–39. IEEE.

Research, C. I. (2020). What is immunotherapy's relationship to the immune system? https://www.cancerresearch.org/immunotherapy/what-is-immunotherapy/.

Scornet, E., Biau, G., Vert, J.-P., et al. (2015). Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741.

Tekpli, X., Lien, T., Røssevold, A. H., Nebdal, D., Borgen, E., Ohnstad, H. O., Kyte, J. A., Vallon-Christersson, J., Fongaard, M., Due, E. U., et al. (2019). An independent poor-prognosis subtype of breast cancer defined by a distinct tumor immune microenvironment. *Nature Communications*, 10(1):1–14.

Yoo, I., Alafaireet, P., Marinov, M., Pena-Hernandez, K., Gopidi, R., Chang, J.-F., and Hua, L. (2012). Data mining in healthcare and biomedicine: a survey of the literature. *Journal of medical systems*, 36(4):2431–2448.