Técnicas de Aprendizado de Máquina na Diagnóstico do Câncer de Mama

Vinícius Pacheco Vieira[1], **Mirkos Ortiz Martins**[1]

[1]Curso de Sistemas de Informação - Centro Universitário Franciscano

vinipachecov@gmail.com, mirkos@unifra.br

Abstract. *This paper describes the development of a machine learning application by using Python programming language and it's assets for pattern matching a group of classified breast cancer data base divided between malignant and benign. The objetive of the algorithm is the classification of the tumor data and allow the insertion of a new tumor data between the classified ones and predict the diagnosis with the application. It was used a deep neural network from the deep learning technique with Google's framework Tensorflow and a Support Vector Machine(SVM) from scikit-learn library. The final accuracy of SVM the and the deep neural network was respectively, 97% and 0.96% for a database with 398 diagnosis used for the training dataset and 171 for test dataset. The dataset used was provided by the Wisconsin University from real tumors, which allows us to conclude that the software can help pathologysts for better diagnosis of malignant breast cancer tumors.*

Resumo. *Este trabalho descreve o desenvolvimento de uma pesquisa de machine learning ou data science, utilizando a linguagem de programação Python, para o reconhecimento de padrões em um conjunto de diagnósticos de câncer de mama classificados entre benignos e malignos. O objetivo foi comparar a implementação de dois algoritmos: redes neurais e máquainas de vetores de suporte, onde o resultado alcançado foi, respectivamente, de 97% e 96% de eficácia. A base utilizada possui 398 diagnósticos usados como treinamento para o algoritmo e 171 diagnósticos para a fase de classificação - o que sugere que o uso de aprendizado de máquina pode auxiliar no trabalho de classificação de tumores de câncer de mama entre as categorias benignas e malignas.*

1. Introdução

O câncer de mama é o tipo que mais atinge as mulheres segundo [inc 2017]. Segundo o estudo, cerca de 28% dos novos casos de câncer todos os anos correspondem a exemplares de câncer de mama. Em 2013, o número de mortes relacionadas ao câncer de mama correspondeu a 14.388, sendo 181 homens e 14.206 mulheres ainda segundo [inc 2017].

Essa doença atinge, geralmente públicos na faixa de 35 anos ou mais, evoluindo suas chances de ocorrência gradualmente com o avanço da idade segundo [inc 2017]. Com o envelhecimento presenciado nas sociedades modernas, realizar pesquisas para auxiliar no diagnóstico precoce de doenças como o câncer de mama, mostram-se investimentos fundamentais para a saúde pública.

Diante desse cenário, muito se tem pesquisado em como introduzir métodos para aumentar a eficácia e diagnosticar mais precocemente a doença. Estudos como [Melo et al. 2014] mostram como algoritmos de aprendizado de máquina podem auxiliar no diagnóstico do câncer de mama atráves da análise de mamografias. As taxas de acurácia chegaram a 80% em alguns algoritmos como em redes neurais artificiais de arquitetura *Multi-Layer-Perceptron (MLP)*.

Com base neste domínio e sabendo da performance do algoritmo de redes neurais utilizado em [Melo et al. 2014], este trabalho pretende comparar algumas técnicas de aprendizado de máquina na avaliação de dados do câncer de mama disponibilizados pela Universidade de Wisconsin [1]. Por meio desta análise foram comparadas as implementações e seus resultados relacionados ao tipo de técnica utilizada de aprendizado de máquina.

2. Objetivos

O objetivo geral deste trabalho foi comparar as diferentes implementações e performances dos algoritmos de aprendizado de máquina na classificação de dados de amostras do câncer de mama.

2.1. Objetivos Específicos

Dentre os objetivos específicos desse trabalho, foram identificados:

- Estudar sobre Redes Neurais, principalmente as de modelo *Multi-Layer Perceptron*, para identificar a melhor abordagem para a classificação dos dados;
- Os algoritmos utilizados foram: redes neurais artificiais *Multi-Layer-Perceptron*, redes neurais profundas com o *framework Tensorflow* e *Support Vector Machines* com a implementação da biblioteca *Scikit-Learn*.
- Construir um modelo de Rede Neural para classificação dos tumores;
- Fazer uso da Linguagem Python para construção do modelo de classificação com o *framework Tensorflow* da Google, como base para construção do modelo de classificação;
- Fazer uso dos dados de amostras do câncer de mama disponibilizados pela Universidade de Wisconsin;

3. Referencial teórico

Para a construção de uma aplicação que auxilie na predição de diagnóstico de câncer de mama, conforme explicado na introdução desse trabalho, foi proposta uma abordagem na área de inteligência artificial, mais especificamente em *machine learning* os quais serão apresentados a seguir. Também será explicado como estão estruturadas as informações dos exames laboratoriais usados como base de treinamento para o software desenvolvido.

3.1. Inteligência Artificial

Inteligência Artificial é uma área que se baseia no conceito de agente inteligente. Segundo [Russell et al. 1995] o grande problema que alimenta a IA é descrever e construir agentes que recebam e percebam aspectos do ambiente e consigam agir com base nesses aspectos.

[1]Base de dados de diagnósticos de câncer de mama: `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\%28Diagnostic\%29`

Cada agente é implementado por uma função que transforma esses estímulos do ambiente para ações, e dentro desse campo de estudo existem diferentes formas de se representar essas funções como *production systems*, *reactive agents*, *logical planners*, *neural networks* e *decision-theoric systems*. Já o aprendizado de máquina é um segmento da área de Inteligência Artificial que se utiliza de métodos estatísticos aliados ao poder computacional para desenvolver algoritmos e estratégias no reconhecimento de padrões de uma determinada amostra *dataset*. Com base neste reconhecimento, é possível inferir novos dados com base nas mesmas variáveis que foram dispostas dos dados utilizados para se chegar aos padrões e assim verificar o grau de acurácia com que esse treinamento foi feito. É importante perceber que não o computador jamais aprende algo, mas apenas calibra pesos referentes a uma equação para aproximar a um grupo de determinados resultados de uma amostra. Dessa forma o que o chamado aprendizado de máquina realiza é uma calibragem lógica para aumentar a acurácia das suas proposições iniciais.

Conforme [de Carvalho 1997] a lógica é um discurso que se propõe a validar algo perante uma premissa anterior, ou seja, recriar o nexo com a frase inicial de um silogismo. O mesmo é feito junto do aprendizado, quando se utiliza uma equação e seus pesos (premissa) para se verificar um determinado resultado (conclusão). Não é possível dizer que se chega mais perto da verdade por simplesmente refinar o elo lógico dentro de uma determinada premissa, pois a lógica não se propõe a isso, mas de apenas recriar o seu nexo discursivo entre premissa e conclusão.

Dessa forma, a calibragem de pesos decorrente dos processamentos em lote feitos no treinamento dos algoritmos de aprendizado de máquina, de fato, não demonstram que a máquina aprendem nada, mas aperfeiçoam suas premissas para se adequarem as conclusões pré-determinadas. Apesar de em publicações como a de [Mitchell et al. 2015] onde se propõe um software que nunca parará de aprender, como uma tentativa de saciar as ambições humanas, a pergunta segue ainda sobre se o que a máquina supostamente absorve ou se o processo empreendido no aprendizado de máquina pode ser considerado um aprendizado em qualquer sentido dessa palavra ou apenas uma imagem metafórica para o que é visto na realidade em todos os seres vivos em maior ou menor grau.

3.2. Classificação de Dados

O campo de *machine learning* é uma área da inteligência artificial que engloba um conjunto de problemas definidos como classificação de dados. Este trabalho seguiu uma métrica para atingir as metas e os objetivos descritos na introdução, organizado conforme o fluxograma mostrado na Figura 1, com os passos previamente elaborados e seguidos durante a implementação.

Após a coleta dos dados, na primeira etapa é feita a classificação dos dados contidos em um arquivo em formato de texto separados por vírgula, `.csv`. A segunda etapa necessária é efetuar um pré-processamento de eliminação de valores inconsistentes, como caracteres ao invés de números para os campos de valores numéricos e espaços vazios, que devem ser preenchidos com o valor zero.

Na existência de entradas de dados com variáveis incompletas, são possíveis duas escolhas: retirar o exemplar ou atribuir algum valor à variável. No caso de o *dataset* ter um tamanho pequeno, a retirada pode comprometer o equilíbrio entre as classes a serem classificadas, por conter alguma informação importante.

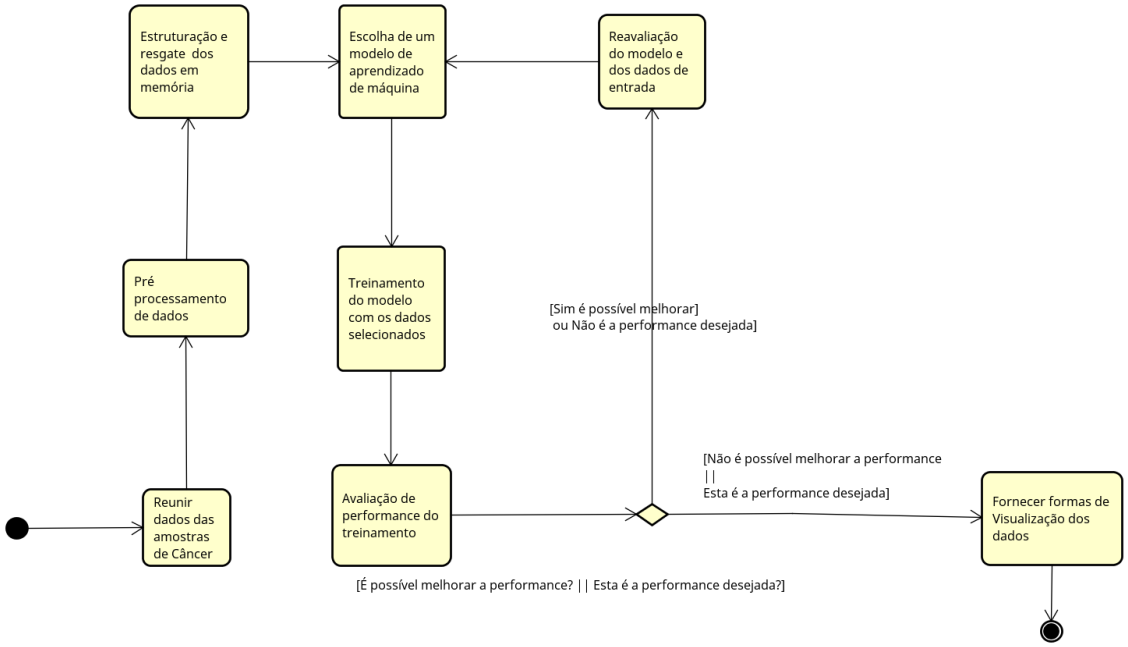

Figura 1. Sequência de processos para implementação de *machine learning*

Nesse caso, seria possível fazer a média dos valores da coluna na qual o dado faltante se encontra e preenchê-la com o resultado do cálculo. Outra saída possível é desfazer-se de dados faltantes sem comprometer muito o balanço entre classes, o que não impede de também se adotar a primeira opção, a de calcular a média, ou outros cálculos para preencher com o resultado.

Nesse trabalho por não haver variáveis sem valores, não foi necessário esse tipo de processamento.

3.3. A base de dados de diagnósticos de câncer de mama

Para esse trabalho, foi utilizada uma base de informações fornecida por um estudo de diagnósticos de câncer de mama, categorizados entre malignos e benignos. As características são calculadas a partir de uma imagem digitalizada de uma amostra aspirada com agulha fina (FNA - *Fine Needle Aspiration*) de uma massa celular coletada no peito da paciente. Os autores do trabalho descrevem as características dos núcleos celulares presentes na imagem (fatiada e de espaço tridimensional)descrito em [Mangasarian et al. 1989]. Este banco de dados também está disponível através do servidor WDBC (*Wisconsin Data Base Cancer*)[2] A base de dados é formulada em uma planilha contendo 34 colunas sendo a primera o ID do paciente, a segunda coluna sendo o diagnóstico (M = maligno, B = benigno) e as outras 32 restantes, características reais do núcleo celular coletado por FNA.

A média, erro padrão e ”pior”ou maior (média dos três maiores valores) dessas características foram calculadas para cada imagem, resultando em 30 recursos. Por exemplo, o campo 3 é Radius Médio, o campo 13 é Radius SE, o campo 3 é o Pior Raio. Todos

[2]WDBC: `UWCS:ftp.cs.wisc.educdmath-prog/cpo-dataset/machine-learn/WDBC/`

os valores das características são recodificados com quatro dígitos significativos.

A planilha possui uma distribuição de classes formada por 357 diagnósticos de câncer benigno e 212 diagnósticos de câncer maligno.

3.4. Trabalhos Correlatos

Trabalhos como de [Becker et al. 2017] mostram a importância e o alto grau de relevância de pesquisas na classificação de tumores utilizando as técnicas de aprendizado de máquina. No estudo, foram utilizadas redes neurais artificiais convolucionais em aprendizado profundo, chegando à conclusão que a acurácia geral de classificação de tumores do câncer de mama por mamografias era a mesma dos radiologistas.

Dentro de uma outra perspectiva, no trabalho de [Wang et al. 2016] foi feito um estudo semelhante com aprendizado profundo, porém para classificação de câncer de mama metastático por meio de imagens. Os resultados do trabalho resultaram em prêmios e a impressionantes indicadores como 0.925 AUC de área abaixo da curva com base no gráfico ROC para imagens inteiras, lembrando que 1.0 é significaria que o classificador é perfeito para identificar a doença. O classificador AUROC (*Area Under the Curve of Receiver Operating Characteristic*), que é uma métrica, conforme [Bradley 1997], que combina as métricas de taxa de verdadeiros positivos e de falsos positivos dentro das predições, variando de 0.5, para uma predição randômica e 1.0 em um classificador perfeito.

Ainda no estudo de [Wang et al. 2016] foi visto que patologistas da área chegavam ao resultado de 0.966 AUC e que ao se utilizarem da ferramenta, alcançaram a 0.995 AUC. Esse valor representa uma redução de 85% dos erros humanos de diagnósticos, reforçando a importância e relevância das pesquisas e ferramentas que podem auxiliar no aumento da acurácia dos diagnósticos dos patologistas.

4. Metodologia

O processo de descoberta de conhecimento conforme [Han et al. 2011] é dividido em 7 etapas. Foi feito da mesma forma, porém com a seleção dos dados sem um *datawarehouse* pois os dados já estão previamente selecionados e organizados. Os outros passos do processo de descoberta de conhecimento foram seguidos e estão representados no diagrama da figura 1.

5. Desenvolvimento e implementação

O protótipo desenvolvido foi estruturado em linguagem Python, implementado com a metodologia *Cowboy*. Python é uma linguagem de programação interpretada e de alto nível [Menezes 2014], e seu propósito é de uso geral [Hetland 2005]. Segundo [Python 2009] a linguagem busca combinar um poder elevado de implementação com uma sintaxe clara. Não bastando isso, possui uma grande biblioteca padrão, facilmente compreensível e documentada [Grus 2016].

Conforme[Summerfield 2010], Python é uma linguagem de fácil aprendizagem e bem difundida, justamente por ser multiplataforma e por ser de fácil leitura, juntamente com sua capacidade de desenvolver aplicações com menos linhas de código do que aplicações equivalentes em C++ ou Java. Além de contar com uma biblioteca padrão

muito completa, em Python, pode-se programar com o paradigma procedural orientado a objeto e, até mesmo, com o paradigma funcional. Vale ressaltar que Python é uma das linguagens mais utilizadas no meio de *machine learning*, sendo, assim, a linguagem com grande quantidade de exemplos e comunidade forte para a área que engloba este trabalho.

5.1. Tratamento dos dados

Os dados coletados estão disponíveis pelo repositório da Universidade de Wisconsin pelo site [University 2017], o qual também constam as suas descrições. Vale ressaltar que os dados foram selecionados por médicos o que enfatiza a sua significância de uso para este trabalho.

Tratamento de dados Categóricos

Em ambos os algoritmos testados foi utilizado o mesmo grupo de dados sob o mesmo processo de pré-processamento, conforme o segundo passo workflow definido na figura 1. Na figura 2 é revelado o processo de leitura com a função *read_csv* da biblioteca *pandas*. Logo em seguida foi feita a visualização dos dados sem tratamento.

```
In [18]:
1 cancer_data = pd.read_csv('data.csv')
2 cancer_data[60:72]
```

Out[18]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_m
60	858970	B	10.170	14.88	64.55	311.9	0.11340	0.08061	0.01084	0.012
61	858981	B	8.598	20.98	54.66	221.8	0.12430	0.08963	0.03000	0.009
62	858986	M	14.250	22.15	96.42	645.7	0.10490	0.20080	0.21350	0.086
63	859196	B	9.173	13.86	59.20	260.9	0.07721	0.08751	0.05988	0.021
64	85922302	M	12.680	23.84	82.69	499.0	0.11220	0.12620	0.11280	0.068
65	859283	M	14.780	23.94	97.40	668.3	0.11720	0.14790	0.12670	0.090
66	859464	B	9.465	21.01	60.11	269.4	0.10440	0.07773	0.02172	0.015
67	859465	B	11.310	19.04	71.80	394.1	0.08139	0.04701	0.03709	0.022
68	859471	B	9.029	17.33	58.79	250.5	0.10660	0.14130	0.31300	0.043
69	859487	B	12.780	16.49	81.37	502.5	0.09831	0.05234	0.03653	0.028
70	859575	M	18.940	21.31	123.60	1130.0	0.09009	0.10290	0.10800	0.079
71	859711	B	8.888	14.64	58.79	244.0	0.09783	0.15310	0.08606	0.028

12 rows × 32 columns

Figura 2. Leitura de Dados e visualização de parte do Dataset

Com a leitura completa dos dados do *dataset* foi feito o tratamento de dados categóricos, pois a coluna 'diagnosis' possui valores 'M' para maligno e 'B' para benigno e não se pode fazer o treinamento dos algoritmos com tais valores. Para isso foi feita uma função que está na figura 3 que transforma os valores 'M' em 1 e 'B' em 0.

Assim tumores malignos ganham o valor 1 e beningnos o valor 0 na coluna diagnosis conforme pode se ver na figura na tabela da 3.

Tratamento de dados Numéricos

Ainda no passo de pré-processamento da figura 1 foi feito também o tratamento de dados numéricos selecionando as colunas a serem normalizadas, segue a figura 4:

Como resultado, tem-se o seguinte grupo de dados conforme a figura 5:

```
In [26]: 1 cancer_data['diagnosis'] = cancer_data['diagnosis'].apply(M_B_0_1)

In [27]: 1 cancer_data[60:72]
```

Out[27]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_mea
60	858970	0	10.170	14.88	64.55	311.9	0.11340	0.08061	0.01084	0.0129(
61	858981	0	8.598	20.98	54.66	221.8	0.12430	0.08963	0.03000	0.0092!
62	858986	1	14.250	22.15	96.42	645.7	0.10490	0.20080	0.21350	0.0865:
63	859196	0	9.173	13.86	59.20	260.9	0.07721	0.08751	0.05988	0.0218(
64	85922302	1	12.680	23.84	82.69	499.0	0.11220	0.12620	0.11280	0.0687:
65	859283	1	14.780	23.94	97.40	668.3	0.11720	0.14790	0.12670	0.0902!
66	859464	0	9.465	21.01	60.11	269.4	0.10440	0.07773	0.02172	0.0150-
67	859465	0	11.310	19.04	71.80	394.1	0.08139	0.04701	0.03709	0.0223(
68	859471	0	9.029	17.33	58.79	250.5	0.10660	0.14130	0.31300	0.0437!
69	859487	0	12.780	16.49	81.37	502.5	0.09831	0.05234	0.03653	0.0286-
70	859575	1	18.940	21.31	123.60	1130.0	0.09009	0.10290	0.10800	0.0795:
71	859711	0	8.888	14.64	58.79	244.0	0.09783	0.15310	0.08606	0.0287:

12 rows × 32 columns

Figura 3. Função de tratamento dos dados categóricos

```
cols_normalizar = ['radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave_points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave_points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave_points_worst',
       'symmetry_worst', 'fractal_dimension_worst']
```

Figura 4. Seleção das colunas a serem padronizadas

```
In [10]: 1 cancer_data[cols_normalizar] = cancer_data[cols_normalizar].apply(lambda x: (x - np.mean(x)) / (np.std(x)  ) )

In [11]: 1 cancer_data
```

Out[11]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_r
0	842302	1	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.53
1	842517	1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.54
2	84300903	1	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.03
3	84348301	1	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.45
4	84358402	1	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.42
5	843786	1	-0.476375	-0.835335	-0.387148	-0.505650	2.237421	1.244335	0.866302	0.82
6	844359	1	1.170908	0.160649	1.138125	1.095295	-0.123136	0.088295	0.300072	0.64

Figura 5. Dataset tratado

5.2. Rede Neural Profunda com Tensorflow

Nesta seção tratar-se-á da implementação em python da rede neural profunda com o *framework Tensorflow*.

5.2.1. Dependências do projeto e Bibliotecas

Foi iniciado o processo pelas importações de bibliotecas necessárias e também a leitura dos dados do *dataset* no arquivo 'data.csv':

Na figura 6 consta a importação das bibliotecas pandas, numpy e tensorflow. Foram utilizados apelidos (alias) para cada uma delas, sendo respectivamente pd, np e tf. A biblioteca pandas é utilizada para abrir arquivos de *datasets*. Já a biblioteca numpy para calculo numérico e algebra linear e por último a bilioteca tensorflow para os algoritmos

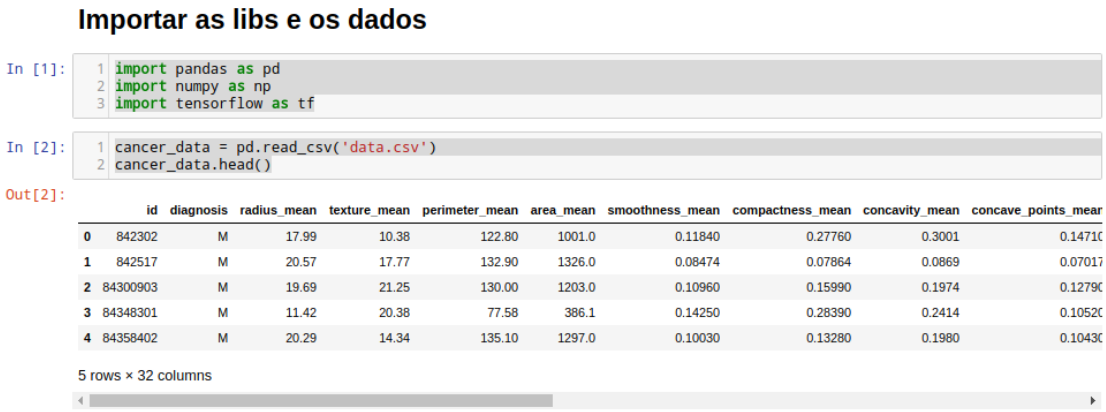

Figura 6. Imports de bibliotecas e leitura de dados

de aprendizado de máquina. Ao final da figura é mostrado como o *dataset* está ao ser recém carregado.

5.2.2. Tratamento dos dados no Tensorflow

Após o tratamento da coluna 'dianogis' é visto quais os nomes de cada coluna e para uma delas é criada uma variável reconhecível pelo tensorflow, lembrando que precisam ser do mesmo tipo, por isso é utilizada o tipo $tf.feature_column.numeric_column$ que é equivalente para tipos numéricos no *Tensorflow*. Na figura 7, consta essa transformação e a retirada da coluna ID pois ela não possui relevância ou correlação com os dados amostrais, tratando-se apenas de identificador. Também é feita a divisão dos dados entre valores de entrada e resultados, sendo cada um guardados nas variáveis `x_data` e `y_data`.

5.2.3. Separação dos datasets de treino e teste

Na figura 8 são feitas as divisões de *datasets* entre treino e testes, para que o modelo treinado seja testado com valores que o mesmo desconhece e assim tenha métricas mais confiáveis nos resultados.

Conforme está na figura 8, é feito uma importação do método *train_test_split* da biblioteca scikit-learn para efetuar a separação. No método, colocar-se-á 70% do *dataset* original para treinamentos e 30% para testes.

5.2.4. Criação do modelo de Rede Neural e vinculação com os dados de treino

Em sequência, na figura 9 foi criado a função para receber os dados dos valores criados no tensorflow, a qual se utilizará para treinar o modelo criado de rede neural profunda.

Na variável *input_func*, é atribuído o valor da equação de treino do *Tensorflow*, misturando a ordem os dados e colocando em grupos de 10 amostras por lote (*batches*). A variavel model receberá o modelo de rede neural profunda, que corresponde a classe

Inserir dados no tensorflow

In [12]:
```
cancer_data.columns
```

Out[12]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave_points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave_points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave_points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

In [13]:
```
radius_mean =tf.feature_column.numeric_column('radius_mean')
texture_mean =tf.feature_column.numeric_column('texture_mean')
perimeter_mean =tf.feature_column.numeric_column('perimeter_mean')
area_mean =tf.feature_column.numeric_column('area_mean')
smoothness_mean =tf.feature_column.numeric_column('smoothness_mean')
compactness_mean =tf.feature_column.numeric_column('compactness_mean')
concavity_mean =tf.feature_column.numeric_column('concavity_mean')
concave_points_mean =tf.feature_column.numeric_column('concave_points_mean')
symmetry_mean =tf.feature_column.numeric_column('symmetry_mean')
fractal_dimension_mean =tf.feature_column.numeric_column('fractal_dimension_mean')
radius_se =tf.feature_column.numeric_column('radius_se')
texture_se =tf.feature_column.numeric_column('texture_se')
perimeter_se =tf.feature_column.numeric_column('perimeter_se')
area_se =tf.feature_column.numeric_column('area_se')
smoothness_se =tf.feature_column.numeric_column('smoothness_se')
compactness_se =tf.feature_column.numeric_column('compactness_se')
concavity_se =tf.feature_column.numeric_column('concavity_se')
concave_points_se =tf.feature_column.numeric_column('concave_points_se')
symmetry_se =tf.feature_column.numeric_column('symmetry_se')
fractal_dimension_se =tf.feature_column.numeric_column('fractal_dimension_se')
radius_worst =tf.feature_column.numeric_column('radius_worst')
texture_worst =tf.feature_column.numeric_column('texture_worst')
perimeter_worst =tf.feature_column.numeric_column('perimeter_worst')
area_worst =tf.feature_column.numeric_column('area_worst')
smoothness_worst =tf.feature_column.numeric_column('smoothness_worst')
compactness_worst =tf.feature_column.numeric_column('compactness_worst')
concavity_worst =tf.feature_column.numeric_column('concavity_worst')
concave_points_worst =tf.feature_column.numeric_column('concave_points_worst')
symmetry_worst =tf.feature_column.numeric_column('symmetry_worst')
fractal_dimension_worst =tf.feature_column.numeric_column('fractal_dimension_worst')
```

In [14]:
```
feat_cols = [radius_mean,texture_mean,perimeter_mean,area_mean,smoothness_mean,compactness_mean,concavity_mean,
             concave_points_mean,symmetry_mean,fractal_dimension_mean,radius_se,texture_se,perimeter_se,area_se,
             smoothness_se,compactness_se,concavity_se,concave_points_se,symmetry_se,fractal_dimension_se,
             radius_worst,texture_worst,perimeter_worst,area_worst,smoothness_worst,compactness_worst,concavity_wo
             concave_points_worst,symmetry_worst,fractal_dimension_worst]
```

Retirar a coluna ID e separar o input do output (coluna diagnosis)

In [15]:
```
labels_to_drop = ['id','diagnosis']
x_data = cancer_data.drop(labels=labels_to_drop,axis=1)
y_data = cancer_data['diagnosis']
```

Figura 7. Inserção das colunas no tensorflow

Separando os datasets de treino e testes.

In [43]:
```
from sklearn.model_selection import train_test_split
```

In [44]:
```
X_train, X_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=101)
```

Figura 8. Separação de dataset de treino e de testes

DNNClassifier do *Tensorflow*. A rede neural possuirá 3 camadas intermediárias de 200 neurônios. No último comando é feito o treinamento de 1000 vezes do modelo pelo *dataset*, atribuindo a função que no início.

5.2.5. Avaliando a Rede Neural Profunda

Para o treinamento foi utilizado os *datasets* de treino *X_train* e *y_train*. Para os testes e a avaliação ou `evaluation` se utilizará os *datasets X_test* e *y_test*. Na figura a 10 tem-se o exemplo da função de avaliação sendo criada e dos resultados obtidos pelo teste:

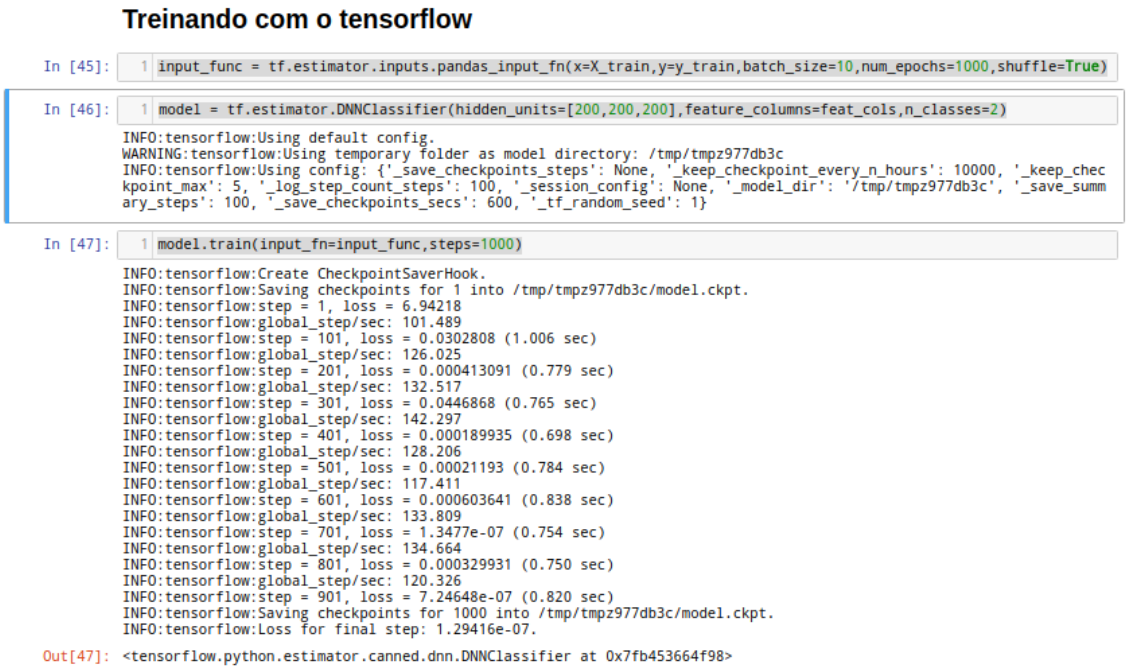

Figura 9. Separação de dataset de treino e de testes

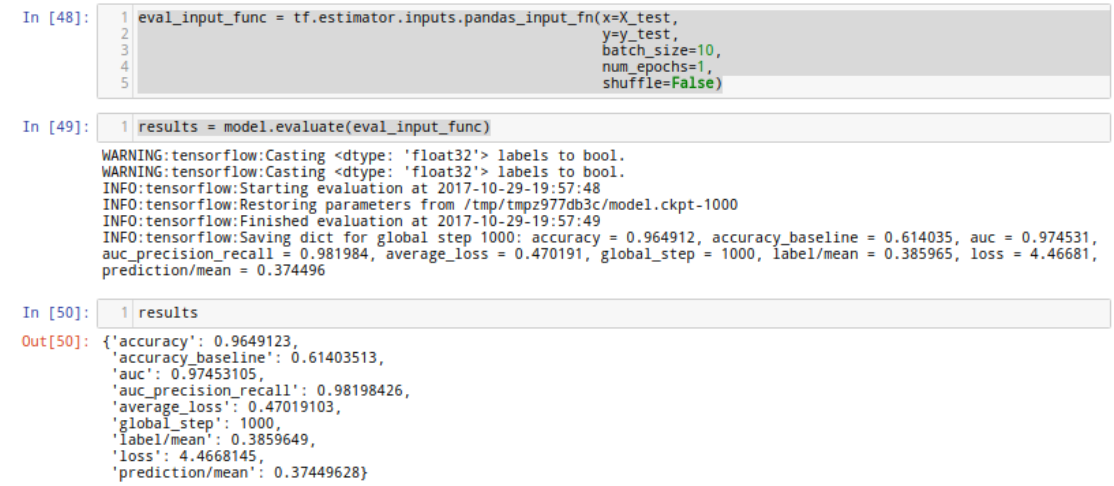

Figura 10. Avaliação do Modelo

Os resultados apontam para 96% de acurácia com base no *dataset* de testes, ou seja, valores com os quais ele não foi treinado.

5.3. Implementação da SVM

Nesta seção tratar-se-á da implementação da SVM e os tratamentos de dados relacionados a essa implementação.

5.3.1. Importação de bibliotecas e dependências

Para desenvolver a implementação da máquina de suporte de vetores, foram utilizadas também as bibliotecas pandas para abertura de arquivos com os dados disponibilizados, *numpy* para álgebra e cálculos diversos e algumas funções e classes da *scikit-learn* para divisão de *datasets* e para instanciar a *SVM* conforme a figura 11.

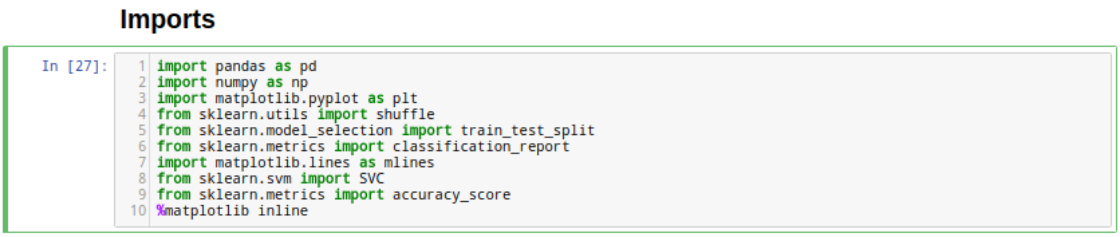

Figura 11. Bibliotecas importadas do sistema

5.3.2. Separando colunas dos dados e resultados

Conforme foi feito também na rede neural profunda, os dados das propriedades dos tumores foram alocados na variável *x_data* e os seus respectivos diagnósticos na variável *y_data*. Também foi retirada a coluna 'id' pois ela não influencia na correlação dos tumores e seus diagnósticos conforme a figura 12

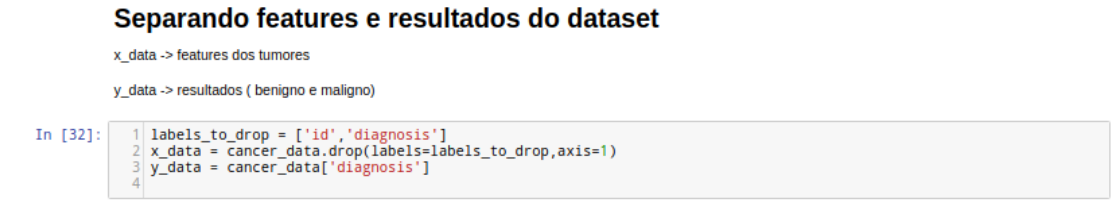

Figura 12. Separando propriedades e diagnósticos do dataset

5.3.3. Separação de Datasets de Treino e Testes

De forma semelhante foi utilizado também a função `train_test_split` da biblioteca *scikit-learn* para separar o *dataset* original em dois *datasets*, um para treino do modelo e outro para testes conforme a figura 13:

Dessa separação criam-se dois *datasets*, o de treino representado pelas variáveis `X_train` e `y_train` e o *dataset* de testes representado por `X_test` e `y_test`.

5.3.4. Criando Classificador SVM

No próximo passo foi instanciada a classe SVC com o *kernel* 'rbf' que é um dos parâmetros de criação da SVM. Em seguida é chamada a função `fit` para treinar nosso

Fazendo o split do Dataset entre dataset de treino e de testes

X_train -> dataset de treino

y_train -> resultados do dataset de treino

X_test -> dataset de testes

y_test -> resultados do dataset de testes

```
In [33]:
1 X_train, X_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=101)
```

Figura 13. Separação do dataset de treino e testes

modelo de SVM com os dados de treino conforme a figura 14:

Criando o classificador SVM e fazendo a classificação dos dados

```
In [34]:
1
2 classifier = SVC(kernel = 'rbf', random_state = 0)
3 classifier.fit(X_train, y_train)

Out[34]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=0, shrinking=True,
  tol=0.001, verbose=False)

In [35]:
1 y_pred = classifier.predict(X_test)
```

Figura 14. Criando classificador e fazendo teste de predição

Ao final, é feito uma predição com os dados de teste em `X_test` dentro da função `predict` da classe `SVC`, armazenando os resultados da predição na variável `y_pred`.

5.3.5. Imprimindo valores dos resultados

Na impressão dos valores de resultado, foi utilizada a função `classification_report` e a função `accuracy_score`, ambas da biblioteca scikit-learn.

Imprimindo valores e resultados

```
In [37]:
1 print(classification_report(y_test,y_pred=y_pred))
2 print('Final Accuracy:',accuracy_score(y_test,y_pred))

             precision    recall  f1-score   support

          0       0.97      0.99      0.98       105
          1       0.98      0.95      0.97        66

avg / total       0.98      0.98      0.98       171

Final Accuracy: 0.976608187135

In [12]:
1 print(accuracy_score(y_test,y_pred))

0.976608187135
```

Figura 15. Resultados da SVM

O resultado final da classificação foi de 0.9766 ou 97.66% de acurácia e 0.98 de média de f1-score. Os altos índices de desempenho corroboram que esse método possui uma boa validade de uso para *datasets* com este perfil.

6. Resultados

Nesta seção serão descritos os resultados obtidos ao final da classificação, treinamento e execução do algoritmo desenvolvido para a preempção de diagnósticos e serão discutidos os seus significados. Na tabela 1 estão os resultados da eficácia dos algoritmos com relação aos seus *datasets* de testes:

Algoritmo Utilizado	Acurácia	F1-Score	Tempo de Processamento
Redes Neurais Profundas	94.4%	0.96	14 segundos
SVM	97.66%	0.98	0 segundos

Tabela 1. Resultados das predições

Os resultados alcançados atingiram uma acurácia de 96% e um f1-score de 0.96, ou seja, mesmo com classes não perfeitamente balanceadas no *dataset*, foi possível alcançar um alto grau de desempenho. O tempo de processamento do treino foi de 14 segundos em CPU. Já o algoritmo de SVM se destaca pelo seu baixo tempo de processamento e uma maior acurácia e F1-Score. Na Figura 16 são mostrados os resultados da classificação da rede neural profunda utilizando o framework *tensorflow*:

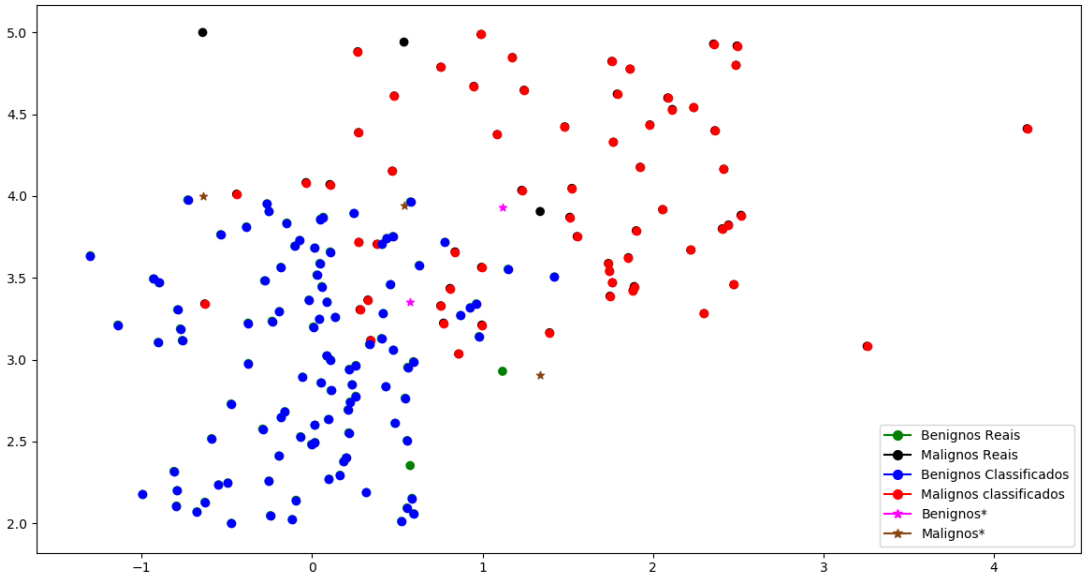

Figura 16. Gráfico das classificações com Redes Neurais Profundas

Para compreender corretamente a precisão que esse gráfico busca representar, deve-se compreender que se houvessem apenas pontos vermelhos e azuis o classificador seria perfeito, ou seja 100% de acertos nos diagnósticos.

Como é possível averiguar na figura 16, existem ainda 2 pontos verdes e duas estrelas rosas, ou seja, tem-se 2 exemplos de tumores que eram benignos, mas foram classificados como malignos. Também, pode-se ver 3 pontos pretos e 3 estrelas douradas, ou seja, 3 tumores malignos que foram classificados como benignos pelo algoritmo.

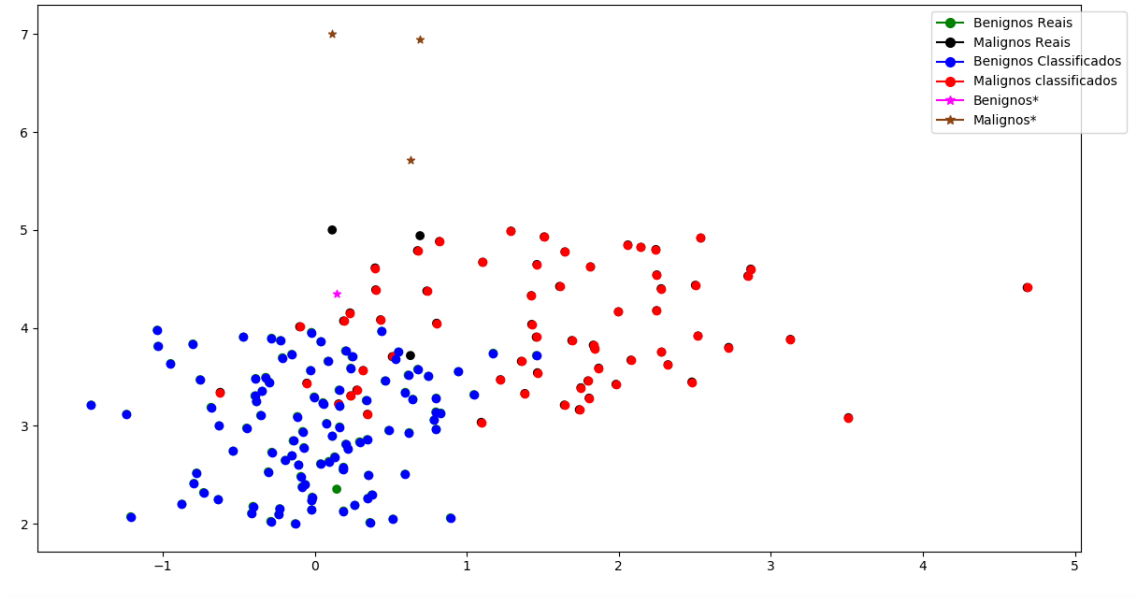

Figura 17. Gráfico das classificações com a SVM

Na Figura 17 são mostrados os resultados obtidos após a execução do algoritmo de classificações do grupo de teste:

De forma semelhante ao gráfico das redes neurais profundas, é possível identificar um ponto verde e uma estrela rosa, o que indica que um tumor benigno foi classificado como maligno. também é possível identificar três pontos pretos e três estrelas douradas, o que significa que três tumores malignos foram classificados como benignos.

7. Conclusões

Neste trabalho foram atingidos os objetivos inicialmente pretendidos, com destaque para o estudo sobre Redes Neurais, principalmente as de modelo *Multi-Layer Perceptron*, para identificar a melhor abordagem para a classificação dos dados, atingindo os objetivos de estudar redes neurais, o que permitiu construir um modelo para classificar tumores com base no *dataset* coletado.

Utilizou-se o *framework Tensorflow* ao invés da API do IBM Watson. Em primeiro lugar, a liberação de acesso e implementação com o IBM Watson não foi possível pelo seguinte motivo: o projeto inicial foi alterado devido a problemas de ordem burocrática do Centro Universitário Franciscano e do departamento de TI para aquisição de dados. Com extrema paciência foi esperado mais de 7 meses e ainda sem resposta, optando por fim pela mudança do tema. A linguagem Python é de grande auxílio para desenvolvimento pois sua rápida curva de aprendizado proporciona com que resultados surjam mais rapidamente trazendo maior estímulo para a pesquisa e mais tempo para problemas não relacionados a implementação. Como resultado final chegou-se a duas implementações de uma pesquisa sobre dois algoritmos capazes de classificar tumores. Chegando a 96% de acurácia, é visível a alta capacidade de classificação que algoritmos de aprendizado de máquina podem desempenhar em atividades como esta que é tão urgente no caso do câncer de mama. Por se tratar de um teste muito dependente do empirismo, compreende-se que as redes neurais profundas tiveram desempenho inferior ao da SVM devido ao

tamanho mais enxuto dos dados analisados.

Mesmo perante a erros, que sim devem ser analisados para tentar alcançar maiores valores de desempenho dos algoritmos, é importante relembrar que trata-se de um sistema de apoio a decisão e não de tomada de decisão.

É evidente que a utilização destas técnicas pelo seu alto grau de eficácia ajudam e devem ser utilizadas como um auxílio no desempenho de diagnósticos médicos do câncer de mama evitando falhas humanas como foi feito em [Wang et al. 2016].A continuação deste trabalho certamente deve ser dado um passo a mais para classificação não apenas de dados dos tumores mas também de classificação de imagens de mamografias e no auxílio de diagnóstico dessas imagens também.

Também é importante citar que devido à problemas burocráticos da área de Tecnologia da Informação, desta Instituição de Ensino deste trabalho, houve o atraso de mais de oito meses na liberação da massa de dados solicitado formalmente através de uma reunião com responsáveis pelo DERCA, setor financeiro e diretor de CPD, o que motivou a mudança do objeto de estudo de classificação de dados de evasão discente (assunto do TFG I) para classificação de tumores do câncer de mama, base essa de livre acesso via internet.

A mudança do assunto principal do projeto ocasionou em mudanças conceituais (a análise dos campos da base de dados teve que ser reescrita e pensada do zero), mudanças tecnológicas (houve um desgaste na procura de solução para conectar um banco de dados relacional e normalizá-lo, trabalho não utilizado nesse outro objetivo, por não usar SGBD), estudar uma nova problemática depois de iniciada a implementação causou toda uma perda em horas-codificação e de levantamento bibliográfico.

Finalmente, que fique registrado, que a falta de diálogo com o detentor das informações necessárias, mesmo dentro da própria Instituição, atrapalhou em muito o desenvolvimento desse trabalho.

Referências

(2017). Instituto nacional de câncer. `http://www2.inca.gov.br/wps/wcm/connect/tiposdecancer/site/home+/mama/cancer_mama`. Accessado em: 28-10-2017.

Becker, A. S., Marcon, M., Ghafoor, S., Wurnig, M. C., Frauenfelder, T., and Boss, A. (2017). Deep learning in mammography: diagnostic accuracy of a multipurpose image analysis software in the detection of breast cancer. *Investigative Radiology*, 52(7):434–440.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

de Carvalho, O. (1997). Aristóteles em nova perspectiva: introdução à teoria dos quatro discursos.

Grus, J. (2016). *Data Science do Zero*. Alta Books.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Hetland, M. L. (2005). *Beginning Python*. Springer.

Mangasarian, O., Setiono, R., and Wolberg, W. (1989). *Pattern recognition via linear programming: theory and application to medical diagnosis*. University of Wisconsin-Madison, Computer Sciences Department.

Melo, M., Gajadhar, A., and Batista, L. V. (2014). Análise comparativa de métodos de aprendizagem de máquina para classificaçao de massas em mamografias. In *Congresso da Sociedade Brasileira de Computaçao-Workshop de Informática Médica*, pages 1772–1775.

Menezes, N. N. C. (2014). *Introdução à programação com python*. Editora Novatec.

Mitchell, T. M., Cohen, W. W., Hruschka Jr, E. R., Talukdar, P. P., Betteridge, J., Carlson, A., Mishra, B. D., Gardner, M., Kisiel, B., Krishnamurthy, J., et al. (2015). Never ending learning. In *AAAI*, pages 2302–2310.

Python, J. (2009). Python (programming language). *Python (programming Language) 1 CPython 13 Python Software Foundation 15*, page 1.

Russell, S., Norvig, P., and Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27.

Summerfield, M. (2010). *Programming in Python 3: a complete introduction to the Python language*. Addison-Wesley Professional.

University, W. (2017). UCI Machine Learning Repository: Breast Cancer Wisconsin (Original) Data Set. `https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)`. [Online; accessed 28-Novembro-2017].

Wang, D., Khosla, A., Gargeya, R., Irshad, H., and Beck, A. H. (2016). Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*.