Nanodegree Engenheiro de Machine Learning

Projeto final

Guilherme Thiago Gomes dos Santos

7 de maio de 2018

I. Definição

Visão geral do projeto

Com a maneira de vida atual da população mundial, existem crescentes necessidades de expansão e controle de ambientes médicos e hospitalares ao redor do mundo. Tal expansão é esperada e necessária, pois vemos cada vez mais um crescente número de pessoas com doenças no coração.

Antigamente, tais problemas eram específicos de uma faixa etária mais avançada, decorrente de problemas inerentes à idade da população, porém vemos que esta situação está mudando, com cada vez mais pessoas de diversas faixas sofrendo de problemas cardíacos.

Mediante esta nova situação, como proceder?

Com a gravidade de um problema no coração, será possível que nós tenhamos uma forma de prever uma fatalidade?

Descrição do problema

Mediante as questões acima, resolvi abordar o tema de problemas cardíacos.

Entrando na área de problemas cardíacos, nós sabemos que existem diversas patologias e síndromes que agem no sistema cardiovascular das pessoas, o que acarreta em problemas no bem-viver e saúde, debilitando o paciente e até podendo traze-lo à óbito.

Dos problemas mais conhecidos e temidos sobre o coração, está o ataque cardíaco. Existem alguns sintomas que alguns pacientes sentem ao estar prestes de um ataque cardíaco, porém, por não ter tido nada, e se sentir ‘imunes’ a problemas cardíacos, o infarto acaba ocorrendo.

Quais são as chances de um determinado paciente que sofreu um ataque cardíaco vir a óbito um ano após o incidente?

Após um ataque, a chance de que uma pessoa venha a óbito eleva-se, pois o coração já sofreu uma lesão, e este paciente precisa de acompanhamento, e é pensando nestes pacientes, que iremos atuar com o projeto.

Métricas

De acordo com o dataset que vamos trabalhar, o nosso problema é de classificação, uma vez que iremos analisar a maior parte dos atributos dos

dados de forma binária, ou seja, se o paciente tem certa característica ou não.

Com isso, utilizaremos de algoritmos de predição para gerar modelos de aprendizado supervisionado, no qual o nosso alvo será se o paciente continuará vivo em um ano após o ataque cardíaco ou se ele vai vir à óbito.

Os algoritmos que utilizaremos serão os seguintes:

- Support Vector Machines (SVM)
- Naive Bayes
- K-N Vizinhos Próximos (K-N Neighbors)

Todos os modelos apresentados serão criados, treinados e testados para verificação de eficiência.

Para tal, utilizaremos o cálculo de F-Score para podermos chegar a um número em que iremos balizar a eficiência de cada um dos modelos preditivos.

Analisarei o tempo gasto por cada uma das atividades do modelo: Tempo gasto para treinamento do modelo, tempo gasto para predição de treino, tempo gasto para predição de teste e o f-score de treino e o de teste.

Após chegarmos a um modelo satisfatório, analisando os números encontrados acima, realizarei um ajuste no algoritmo utilizado para encontrar os melhores parâmetros a serem utilizados na nossa situação.

Em seguida, realizarei uma nova análise para checar os tempos gastos e o f-score, para ver e resultou em alguma melhora de performance.

II. Análise

Exploração dos dados

Os dados(dataset) que eu utilizei para a elaboração deste projeto foram obtidos através do site de Machine Learning da Universidade da California em Irvine (UCI).

As fontes dos dados são:

- Doador: Steven Salzberg

- Coletor dos dados: Dr. Evlin Kinney (The Reed Institute)

O dataset possui dados relevantes sobre pacientes que sofreram um ataque cardíaco de diversos tipos como:

- Survival;
- Still alive;
- age at heart attack;
- pericardial effusion;
- fractional shortening;
- epss;
- lvdd;

- wall motion score;
- wall motion index;

e a nossa coluna alvo:

- alive at 1

No dataset também existem outras colunas como ‘name’ e ‘group’, mas elas serão removidas por questões éticas e também por não fazer parte do contexto da nossa pesquisa.

Por os dados não estarem totalmente completos, ou seja, alguns registros de pacientes não possuírem todos os valores de todos os atributos, foi necessário uma filtragem, para remover valores inválidos e/ou incompletos, para não prejudicar os modelos de aprendizado de máquina na parte de treino, teste e predição.

Como veremos no dicionário abaixo, o dataset utilizado possui diversos termos específicos da àrea, em que ficarão um pouco mais claros ao nosso entendimento após uma breve descrição:

Dicionário dos dados:

Survival: Se o paciente sobreviveu ao incidente.

Still alive: Se o paciente continua vivo.

Age at heart attack: Idade do paciente ao sofrer o ataque cardíaco.

Pericardial effusion (ou efusão pericárdica) : Compressão do coração causada pela coleta de líquido no saco que envolve o coração.

Fractional Shortening (ou fração de encurtamento): Representa a redução percentual do diâmetro cavidade do ventrículo esquerdo durante a ejeção ventricular máxima.

Epss (E point to septal separation): É a distância entre o ponto E em diástole e o septo. Ele é um indicador de aumento do volume ventricular esquerdo.

Lvdd (ou Disfunção diastólica do ventrículo esquerdo): Disfunção diastólica significa que o enchimento ventricular está prejudicado, e a pressão de enchimento final diastólica do ventrículo esquerdo pode estar elevada.

Wall motion Score (ou pontuação do movimento da parede): Índice de ventrículo normocinético.

Wall motion index (ou índice de pontuação do movimento da parede): Também é um índice de ventrículo normocinético.

Alive at 1: Vivo em 1 ano após o ataque cardíaco.

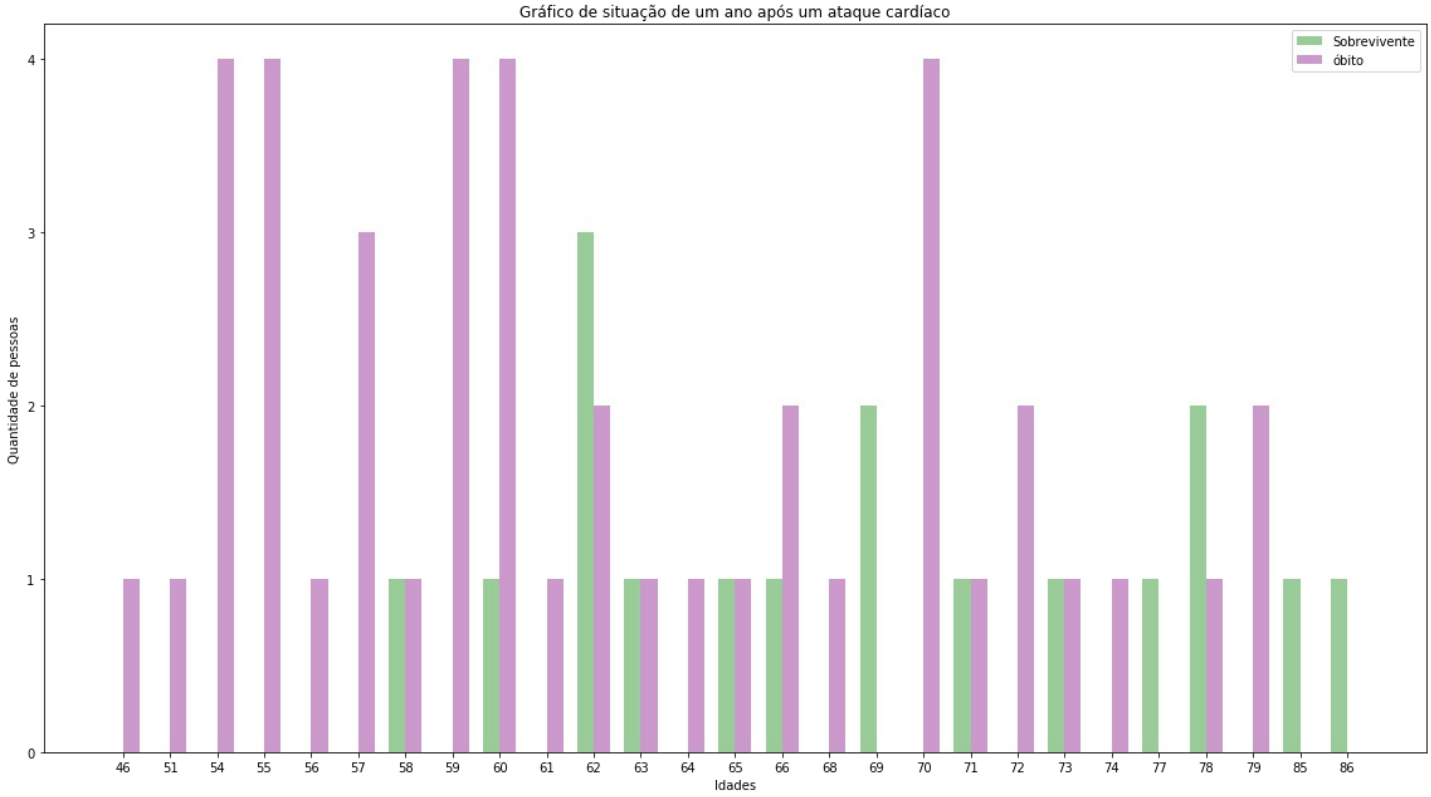
Visualização exploratória

Logo no início da análise, vi a necessidade de sabermos mais informações sobre estes pacientes, algo além de uma mera visualização do dataset

apresentado.

Sobre isso, analisei quais são as idades dos pacientes envolvidos na nossa pesquisa. Também analisei quantos desses pacientes sobreviveram após um ano do incidente e quantos vieram a óbito.

Com isso, cheguei ao seguinte gráfico:



Aqui vemos que de acordo com o dataset, a maior parte dos nossos pacientes vem a óbito em até um ano após o ataque cardíaco.

Não houve nenhum sobrevivente entre os pacientes com idade abaixo dos 57 anos; todos os sobreviventes possuem 58 anos ou mais, com a quantidade de sobreviventes atingindo seu ápice em 62 anos.

Algoritmos e técnicas

Ao escolher o tema do projeto e analisar os metadados do dataset, percebi que a maior parte dos valores é classificatório, ou seja, ele “é ou não é” um determinado atributo, chegando assim na nossa primeira conclusão de escolha de algoritmos.

Após isso, verifiquei também que o que nós temos que atingir é o aumento da sobrevivência dos pacientes, e para isso, precisamos saber quem está correndo risco de vir à óbito, antes que tal fatalidade ocorra. Com os dados em mãos, nós conseguimos fazer esta predição treinando modelos e utilizando a coluna de ‘alive at 1’ para determinar se o paciente sobreviveu ou não.

Este tipo de treinamento, utilizando dados para o modelo ‘estudar’ e determinando que tais dados tem determinada conclusão, são chamados de Aprendizagem Supervisionada, e são estes algoritmos que nós utilizaremos.

Como nós não sabemos qual modelo melhor se adequará ao nosso problema, eu achei melhor testarmos 3 algoritmos, gerando assim três modelos preditivos, e desta forma, através de métricas, analisar qual é o mais eficiente no nosso caso.

Os algoritmos que eu escolhi foram:

- Support Vector Machines (SVM)

- Naive Bayes
- K-N Vizinhos Próximos (K-N Neighbors)

Motivos das escolhas:

Naive Bayes:

- Prós: Por ser um algoritmo fácil de ser preparado para o treino e também por ser um algoritmo rápido.

- Contras: Ele não é eficaz quando é necessário fazer predições utilizando 2 critérios. Ex.: Ao utiliza-lo em um sistema de busca, caso um usuário procure por ataque cardíaco, o modelo irá retornar registros sobre ataques e sobre cardíacos, não sobre ataques-cardíacos.

Support Vector Machines (SVM):

- Prós: Por ser um algoritmo simples e com boa performance em cenários lineares.

- Contras: Não é tão eficiente com grandes quantidades de dados.

K-N Vizinhos próximos:

- Prós: Por não precisar de treinamento e é bem eficiente para uma massa de dados pequena.

- Contras: Ele é um pouco lento na predição.

Tais modelos foram escolhidos com base na simplicidade e também no tamanho do dataset.

No decorrer do processo de criação dos modelos, utilizarei também técnicas de cross-validation para facilitar o uso dos dados.

Também utilizarei cálculo de f-score para auxiliar na conclusão de qual algoritmo escolher.

Após tal escolha, utilizarei GridSearch para realizar um tuning no modelo escolhido, com o objetivo de melhorar a performance dele.

Benchmark

O Algoritmo será escolhi através de alguns critérios:

- Tempo gasto no treino
- Tempo gasto na predição do treino
- F-score do treino
- Tempo gasto no teste
- Tempo gasto na predição do teste
- F-score do teste

Com cada uma dessas informações sobre cada um dos modelos, temos insumos suficientes para chegar a uma conclusão de qual modelo escolher.

III. Metodologia

Pré-processamento de dados

No início, o dataset foi baixado através do site de machine learning da UCI (Universidade da California em Irvine);

Antes da carga inicial, coloquei como primeira linha (linha de cabeçalho) do dataset, o nome da cada coluna, para facilitar a identificação do arquivo csv.

```
In [276]: echo_data = pd.read_csv("echo.csv", na_values='?')
          echo_data = pd.DataFrame(echo_data, index=echo_data.index)
          print "rodou!" # Test ok

          rodou!
```

No parâmetro da primeira linha, eu tive que por o parametro “na_values=’?’” para indicar que os parametros inválidos e ou não preenchidos no dataset serão trocados por NaN (Not a Number).

A princípio, os dados ficaram desta forma:

```
In [277]: display(echo_data.head())
          #display(echo_data)
```

	survival	still-alive	age-at-heart-attack	pericardial-effusion	fractional-shortening	epss	lvdd	wall-motion-score	wall-motion-index	mult	name	group	alive-at-1
0	11.0	0.0	71.0	0.0	0.260	9.000	4.600	14.0	1.00	1.000	name	1.0	0.0
1	19.0	0.0	72.0	0.0	0.380	6.000	4.100	14.0	1.70	0.588	name	1.0	0.0
2	16.0	0.0	55.0	0.0	0.260	4.000	3.420	14.0	1.00	1.000	name	1.0	0.0
3	57.0	0.0	60.0	0.0	0.253	12.062	4.603	16.0	1.45	0.788	name	1.0	0.0
4	19.0	1.0	57.0	0.0	0.160	22.000	5.750	18.0	2.25	0.571	name	1.0	0.0

Abaixo está o código de como foi a formatação de dados, desde remoção das colunas desnecessárias até dos registros inválidos:

```
In [278]: data_format = echo_data.drop(['name', 'mult', 'group'], axis= 1)
          data_format = data_format.replace(np.nan, '0')
          data_format = data_format.drop(data_format[data_format['age-at-heart-attack'] == '0'].index)
          print "data format : {}".format(len(data_format))

          data_format_limpo = echo_data.drop(['name', 'mult', 'group'], axis= 1)
          data_format_limpo = data_format_limpo.dropna(axis=0, how='any')
          print "data format limpo: {}".format(len(data_format_limpo))

          data format : 127
          data format limpo: 61
```

Como consequência da remoção dos dados inválidos, o número de registros válidos para o projeto caiu de 127 para 61 registros.

Implementação

O Código da implementação do projeto encontra-se no mesmo repositório que este documento, sob o nome de “Projeto Ecocardiograma.ipynb”.

Eu utilizei como ferramenta o IDE Jupyter notebook, criando assim um novo notebook exclusivo para o projeto.

Antes de iniciar a criação dos modelos preditivos, elaborei algumas análises e verifiquei alguns dados estatísticos sobre os pacientes e os dados em si:

```
In [280]: #dadosPacientes(data_format)
          dadosPacientes(data_format_limpo)

          Número total de pacientes da triagem: 61 pacientes
          Número total de parametros: 9 parâmetros

          Idade minima dos pacientes: 46 anos
          Idade máxima dos pacientes: 86 anos
          Média de idade dos pacientes: 64 anos

          Número de pacientes sobreviventes ao primeiro ano após o ataque cardíaco: 17 pacientes
          Número de pacientes falecidos ao primeiro ano após o ataque cardíaco: 44 pacientes
          Porcentagem de sobrevivência após ter passado um ano de um ataque cardíaco: 27.87%
```

Como vemos nos valores acima, de um total de 61 pacientes, apenas 17 sobrevivem à um ano após um ataque cardíaco, isto é, 28% do total, e é este valor que precisamos aumentar.

Após isso, eu criei também o gráfico de sobreviventes e óbitos em relação à idade(fig 1).

A próxima etapa é de preparar os dados para a criação dos modelos preditivos. Então para isso, eu dividi os dados em “Atributos gerais” e “Atributo alvo”, tendo o atributo-alvo a coluna alive-at-1.

Após esta etapa, utilizei a técnica de cross validation, e assim eu obtive 4 dataframes, um de treino e um de teste, cada um desses 2 divididos em mais dois: atributos gerais e atributo-alvo.

Em seguida, fiz o código para os 3 modelos preditivos a serem analisados. A estrutura em que criei foi feita de uma forma que facilite o reuso, evitando assim escrever codigo em excesso. Ela realiza as métricas descritas na sessão Benchmarking.

```
In [293]: iterarModelos(clf)

#####
Modelo de predição selecionado: SVC
Tamanho do modelo de treino: 40
Iniciando o treino.
Modelo treinado em 0.0008 segundos
Predição realizada em 0.0003 segundos.
F1 score para o grupo de treino: 1.0.
Predição realizada em 0.0002 segundos.
F1 score para o grupo de teste: 0.75.

#####
Modelo de predição selecionado: KNeighborsClassifier
Tamanho do modelo de treino: 40
Iniciando o treino.
Modelo treinado em 0.0004 segundos
Predição realizada em 0.0007 segundos.
F1 score para o grupo de treino: 1.0.
Predição realizada em 0.0008 segundos.
F1 score para o grupo de teste: 1.0.

#####
Modelo de predição selecionado: GaussianNB
Tamanho do modelo de treino: 40
Iniciando o treino.
Modelo treinado em 0.0008 segundos
Predição realizada em 0.0002 segundos.
F1 score para o grupo de treino: 1.0.
Predição realizada em 0.0003 segundos.
F1 score para o grupo de teste: 1.0.
```

Refinamento

Após a escolha do algoritmo de naive bayes como melhor, me deparei com um problema: O algoritmo de naive bayes não permite tuning, uma vez que não possui parametros.

Como proceder?

Resolvi então realizar o tuning em cima do algoritmo de Support Vector Machines, um algoritmo que teve um desempenho bem interessante, e desta forma, compara-lo novamente com o Naive Bayes, para ver se o tuning gerou uma melhora significativa a ponto de ultrapassar o desempenho do algoritmo escolhido.

Para isso, eu utilizei o algoritmo de GridSearch, para chegar a um tuning ideal do algoritmo de SVM no nosso dataset:

Os atributos de tuning utilizados foram: Kernel, C e Gamma.

- Kernel: São funções utilizadas para transformar os vetores originais, projetand eles em um espaço dimensional elevado, de onde ficará mais fácil de serem separados linearmente.

  Tipos de função Kernel que utilizaremos:
  RBF (Radial Basis Function);
  Linear;
  Poly (Polinomial);

- C: O parâmetro C informa ao modelo de Support Vector Machines quanto você deseja evitar classificar erroneamente cada exemplo de treinamento.
  Valores que utilizaremos: 0.001; 0.01. 0.1; 1; 10;

- Gamma: O Parâmetro Gamma determina a amplitude da função de kernel, que não é influenciada pela direção, e sim, somente pela

distância.

Valores que utilizaremos: 0.001; 0.01. 0.1; 1;

Abaixo está o resultado do grid search com algumas variações dos atributos:

In [294]:
```
parametros = {'kernel':('linear', 'rbf', 'poly'), 'C':[0.001, 0.01, 0.1, 1, 10], 'gamma':[0.001, 0.01, 0.1, 1]
gridSearch = GridSearchCV(svc, parametros)

gridSearch.fit(X_test, np.ravel(y_test))

gridSearch.best_params_
```

Out[294]:
```
{u'C': 0.001, u'gamma': 0.01, u'kernel': u'poly'}
```

Como resultado do tuning, vemos que o algoritmo de Support Vector Machines se comporta melhor com os seguintes parâmetros:

- C: 0.001
- gamma: 0.01
- Kernel: 'poly'

IV. Resultados

Modelo de avaliação e validação

Uma nova iteração entre o algoritmo de SVM ‘tunado’ e o naive bayes foi realizado:

In [295]:

```
svc = SVC(C=0.001, gamma=0.01, kernel='poly')
clf2 = (svc, bayes)

iterarModelos(clf2)
```

```
#####
Modelo de predição selecionado: SVC
Tamanho do modelo de treino: 40
Iniciando o treino.
Modelo treinado em 0.0008 segundos
Predição realizada em 0.0003 segundos.
F1 score para o grupo de treino: 1.0.
Predição realizada em 0.0002 segundos.
F1 score para o grupo de teste: 0.75.

#####
Modelo de predição selecionado: GaussianNB
Tamanho do modelo de treino: 40
Iniciando o treino.
Modelo treinado em 0.0009 segundos
Predição realizada em 0.0003 segundos.
F1 score para o grupo de treino: 1.0.
Predição realizada em 0.0003 segundos.
F1 score para o grupo de teste: 1.0.
```

Como vemos acima, apesar do tuning no modelo de Support Vector Machines, o modelo de Naive Bayes(GaussianNB) continua sendo o que possui a melhor performance.

Após esta última iteração, procurei checar se podemos melhorar a dimensionalidade do dataset, removendo assim colunas desnecessárias.

E com isso, cheguei na seguinte conclusão:

```
Coluna removida: "survival" - Score sem a coluna: 0.0
Coluna removida: "still-alive" - Score sem a coluna: 0.75
Coluna removida: "age-at-heart-attack" - Score sem a coluna: 0.0
Coluna removida: "pericardial-effusion" - Score sem a coluna: 0.95
Coluna removida: "fractional-shortening" - Score sem a coluna: 1.0
Coluna removida: "epss" - Score sem a coluna: 0.15
Coluna removida: "lvdd" - Score sem a coluna: 0.1
Coluna removida: "wall-motion-score" - Score sem a coluna: 0.1
Coluna removida: "wall-motion-index" - Score sem a coluna: 0.1
```

Como podemos observar no resultado acima, dentre os atributos que nos temos no dataset, a maioria deles altera bastante o score. Com a nosso dataset atual, não vejo necessidade de aumento de performance, pois temos poucos registros e isso não está influenciando no processamento, mas caso houvesse problemas relacionados à performance, eu removeria as colunas de 'fractional-shortening' e 'pericardial-effusion', seguindo esta ordem.

Justificativa

Diante das analises realizadas, e dos resultados atingidos, em ambas iterações realizadas, vejo que o modelo de Naive Bayes se adequa perfeitamente para a nossa situação, tanto de performance como de tamanho de dataset.

Os demais modelos também tiveram uma performance interessante, mas na combinação dos fatores entre tempo gasto e f-score, o modelo de Naive Bayes mostrou-se mais interessante.

V. Conclusão

Foma livre de visualização

Com a abordagem que foi tomada no projeto, conseguimos ver através de números e gráfico, as dificuldades de que estão sujeitos os pacientes de problemas cardíacos. Com isso, utilizando os métodos acima elaborados, podemos criar uma situação de cuidados mais intensos para tais pacientes, afim de aumentar a sobrevivência deles.

Reflexão

Creio que com a elaboração do projeto, ele mostrou-se ser de um grande valor para ambientes hospitalares utilizarem como base, tanto em setores que cuidem de problemas cardíacos, como foi o nosso projeto, como que se utilizem como instrumento para criação de idéias semelhantes para diversos outros setores como Oncologia, Pneumologia, entre outros.

Um aspecto que eu achei um tanto desafiador foi pela grande diminuição de registros após o refino dos dados. Com esta operação, os dados do dataset praticamente foram reduzidos pela metade, situação que no meu entendimento, dificultou um pouco.

Melhorias

Um ponto de melhora que eu vejo no projeto foi o tamanho do dataset.

Se o dataset tivesse um volume maior de dados, os valores encontrados seriam mais interessantes, creio que até com um acréscimo de precisão, tanto de score quanto de tempo de treinamento e predição.

---

referências:

MANN, et al. Título: Braunwald Tratado de Doenças Cardiovascular. vol2. Editora: Sauders Elsevier

JUNIOR, Wilson M. Título: Manual De Ecocardiografia. Editora Manole

OTTO; Scwaegler; Freeman. Título: Ecocardiografia guia essencial. Editora Elseviser

BONACCORSO, G. (2017) Machine Learning Algorithms(página 120) , Editora Packt

Capítulo 04 - Avaliação da Função, Ventricular. Disponível em: http://www.bibliomed.com.br/bibliomed/books/livro3/cap/cap04.htm. Acesso em: 8 maio 2018.

Wall motion score index predicts mortality and functional result after surgical ventricular restoration for advanced ischemic heart failure.

Disponível em: https://academic.oup.com/ejcts/article/35/5/847/464717. Acesso em: 8 maio 2018.

Left Ventricular Diastolic Dysfunction (LVDD) & Cardiovascular Autonomic Neuropathy (CAN) in Type 2 Diabetes Mellitus (DM): A Cross-Sectional Clinical Study. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4347109/. Acesso em: 8 maio 2018.

MORAIS - Lídia Gomes de - Papel da Ecocardiografia no diagnóstico da disfunção ventricular esquerda" (Revisão da Literatura). Disponível em: https://www.cetrus.com.br/aluno-artigos/papel-da-ecocardiografia-no-diagnostico-da-disfuncao-ventricular-esquerda-revisao-da-literatura. Acesso em: 8 maio 2018.