Francisco Serôdio Jesus

Preparação de dados e desenvolvimento de aplicação modular para a gestão de hipertensão e diabetes

UNIVERSIDADE D COIMBRA 1 2 9 0

Francisco Serôdio Jesus

Preparação de dados e desenvolvimento de aplicação modular para a gestão de hipertensão e diabetes

Dissertação no âmbito do Mestrado em Engenharia Informática orientada pelos Professores Doutores Jorge Henriques e Marco Simões e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2022

UNIVERSIDADE D
COIMBRA

Francisco Serôdio Jesus

Data Preparation and development of a modular application for hypertension and diabetes management

Dissertation within the scope of the Masters in Computer Engineering supervised by Professor Jorge Henriques and Professor Marco Simões and presented to the Department of Computer Engineering of the Faculty of Science and Technology of the University of Coimbra

September of 2022

O júri

Presidente:

Professor César Alexandre Domingues Teixeira

Professor Associado do Departamento de Engenharia Informática da Universidade de Coimbra

Vogais:

Professor Carlos Nuno Bizarro e Silva Laranjeiro

Professor Auxiliar do Departamento de Engenharia Informática da Universidade de Coimbra.

Professor Jorge Manuel Oliveira Henriques

Professor Associado do Departamento de Engenharia Informática da Universidade de Coimbra

Agradecimentos

Agradeço aos meus à minha família, amigos, companheiros e orientadores o apoio e incentivo necessário para que terminasse esta etapa importante.

Este trabalho foi financiado pelo projeto POWER (número de concessão POCI-01-0247-FEDER-070365), co-financiado pelo European Regional Development Fund (FEDER), através de Portugal (PT2020), e pelo Competitiveness and Internationalization Operational Programme (COMPETE 2020).

Palavras-chave: Pré-processamento de dados, imputação de valores em falta, deteção de anomalias, séries temporais, dados clínicos, pressão sanguínea, glicose

Resumo: A presença de valores em falta e de anomalias são problemas que são comuns e que estão presentes em várias séries temporais. Estes prejudicam alguns métodos e algoritmos de análise de dados, sendo necessário fazer um pré-processamento dos dados que elimine as anomalias e que faça a substituição dos valores em falta por valores razoáveis. O foco desta dissertação é o desenvolvimento e testagem de um módulo de pré-processamento de dados que contenha algoritmos que façam a deteção de anomalias e imputação de valores em falta, em séries temporais com dados de pressão sanguínea e glicose. Foi desenvolvido um estudo, após a seleção de potencias algoritmos para resolverem os problemas expostos, que visa avaliar e posteriormente determinar os melhores métodos. Uma aplicação demonstrativa foi criada para expor o trabalho realizado em conjunto com uma interface a ser entregue para o projeto "POWER - Empowering a digital future".

Keywords: Pre-processing of data, missing value imputation, outlier detection, time series, clinical data, blood pressure, glucose

Abstract: The presence of missing data and outliers are problems that are common and are present in various time series. This problems harm some methods and algorithms of data analysis, so it is necessary to do a pre-processing of the data, in away that eliminates outliers and substitutes the values that are missing for reasonable values. The focus of this dissertation is the development and testing of a data pre-processing module that contains algorithms that do outlier detection and missing value imputation, in time series with data about blood pressure and glucose. It was developed a study, after the selection of potencial algorithms to solve the problems already mentioned, to evaluate and posteriorly determine the best methods. An demonstrative application was created to expose the work done and it was developed an interface to be delivered to the project "POWER - Empowering a digital future".

Índice

Índice 13
Lista de Figuras 16
Lista de Tabelas 18
Lista de Abreviações de Acrónimos 20
Capítulo 1 22
Introdução 22
1.1 Enquadramento 22
1.2 Motivação 23
1.3 Objetivos 23
1.4 Organização do documento 24
Capítulo 2 26
Estado da Arte 26
2.1 Introdução 26
2.2 Séries Temporais 26
2.2.1 Tendência 27
2.2.2 Sazonalidade 27
2.2.3 Ciclos 28
2.2.4 Componente Irregular (Aleatoriedade) 28
2.2.5 Autocorrelação 28
2.2.6 Estacionaridade 28
2.3 Deteção de anomalias 29
2.3.1 Algoritmos estatísticos 30
2.3.2 Algoritmos de clustering 31
2.4 Imputação de dados em falta 32
2.4.1 Imputação Simples 33
2.4.2 Imputação Múltipla 34
Capítulo 3 36
Métodos 36
3.1 Introdução 36
3.2 Bases de dados utilizadas 36
3.2.1 myHeart 36
3.2.2 Dataset da APDP 37
3.3 Algoritmos para deteção de outliers 37
3.3.1 Descrição 37
3.3.1.1 Intervalo Interquartil 37
3.3.1.2 DBSCAN 38
3.3.1.3 Isolation Forest 40
3.3.2 Abordagem para simular anomalias 42
3.3.3 Avaliação 42
3.4 Algoritmos para imputação de dados em falta 43
3.4.1 Descrição 43
3.4.1.1 Imputação simples com média 43
3.4.1.2 MICE 43
3.4.1.3 K-Nearest Neighbours 44
3.4.1.4 MissForest 45
3.4.1.5 VAR-IM 47
3.4.1.6 ARIMA 49
3.4.2 Abordagem para simular dados em falta 50
3.4.3 Avaliação 52
Capítulo 4 54

Aplicações 54
4.1 Introdução 54
4.2 Interface de pré-processamento de dados 54
4.2.1 Especificação da interface 54
4.2.2 Testes 56
4.3 Aplicação demonstrativa 58
4.3.1 Requisitos 58
4.3.2 Arquitetura 62
4.3.3 Interações da aplicação 63
4.4 Conclusão 64
Capítulo 5 66
Resultados e Discussão 66
5.1 Introdução 66
5.2 Dados em falta 66
5.2.1 Resultados 66
5.2.2 Discussão 68
5.3 Outlier 69
5.3.1 Resultados 69
5.3.2 Discussão 71
5.4 Conclusão 71
Capítulo 6 73
Conclusão e trabalho futuro 73
Referências 75
Apêndice A - Especificação da Interface 79
Apêndice B - Testes Realizados 84
Apêndice C - Resultados 89

Lista de Figuras

Figura 3.1: Algoritmo representativo do funcionamento do DBSCAN para deteção de anomalias [23] 39

Figura 3.2: Algoritmo representativo do funcionamento do Isolation Forest durante a fase de treino [26] 40

Figura 3.3: Algoritmo da criação de uma iTree [26] 41

Figura 3.4: Algoritmo do processo do cálculo do tamanho de caminho para cada valor. [26] 42

Figura 3.5: Algoritmo de imputação usando Random Forest [21] 46

Figura 3.6: Esquema representativo do funcionamento do VAR-IM [20] 49

Figura 3.7: Diferentes distribuições possíveis para a determinação de probabilidades para cada peso. [25] 51

Figura 3.8: Esquema representativo do processo de criação de valores em falta [25] 52

Figura 4.1: Diagrama de casos de uso da aplicação demonstrativa 60

Figura 4.2: Arquitetura da aplicação demonstrativa 62

Figura 4.3: Aplicação Demonstrativa 63

Lista de Tabelas

Tabela 4.1: Descrição dos casos de uso da aplicação demonstrativa 61

Tabela 5.1: Resultados da imputação para dados com BP 66

Tabela 5.2: Resultados da imputação para dados com BP, BW, BR e HR 67

Tabela 5.3: Resultados da imputação para dados com BP, BR e HR 67

Tabela 5.4: Resultados da imputação para dados com BP e HR 67

Tabela 5.5: Resultados da imputação para dados com glicose 67

Tabela 5.6: Resultados da deteção de anomalias para dados com BP 69

Tabela 5.7: Resultados da deteção de anomalias para dados com BP, HR, BW e BR 70

Tabela 5.8: Resultados da deteção de anomalias para dados com BP, BR e HR 70

Tabela 5.9: Resultados da deteção de anomalias para dados com BP e HR 70

Tabela 5.10: Resultados da deteção de anomalias para dados com glicose 70

Tabela C.1: Resultados da imputação para dados com BP e BW 89

Tabela C.2: Resultados da imputação para dados com BP e BR 89

Tabela C.3: Resultados da imputação para dados com BP, HR e BW 89

Tabela C.4: Resultados da imputação para dados com BP, BW e BR 90

Tabela C.5: Resultados da deteção de anomalias para dados com BP e BW 90

Tabela C.6: Resultados da deteção de anomalias para dados com BP e BR 90

Tabela C.7: Resultados da deteção de anomalias para dados com BP, HR e BW 90

Tabela C.7: Resultados da deteção de anomalias para dados com BP, BW e BR 90

Lista de Abreviações de Acrónimos

K-NN	K Nearest Neighbours
MICE	Multiple Imputation by Chained Equations
APDP	Associação Protetora dos Diabéticos de Portugal
DBSCAN	Density Based Spatial Clustering of Applications with Noise
VAR-IM	Vector Autorregressive Imputation Method
MCAR	Missing Completely At Random
MAR	Missing At Random
MNAR	Missing Not At Random
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
EM	Expectation-Maximization
PEM	Predictive Error Minimization
BP	Blood Pressure
BW	Body Weight
HR	Heart Rate
BR	Breathing Rate
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives
ROC	Receiver Operating Curve
AUC	Area Under the Curve

Capítulo 1

Introdução

1.1 Enquadramento

O trabalho desenvolvido para esta dissertação enquadra-se no projeto intitulado "POWER - Empowering a digital future", especificamente no sub-projeto 4 "Future Devices".

O projeto POWER é liderado pela Altice Labs, este projeto prevê ainda a importante participação de três entidades do Sistema Nacional de Investigação e Inovação, no papel de co-promotores e parceiros estratégicos nas atividades de investigação e internalização de conhecimento: a Universidade de Coimbra (UC), o Instituto de Telecomunicações de Aveiro (IT Aveiro) e o Instituto Pedro Nunes (IPN).

O presente projeto, intitulado "POWER - Empowering a digital future", visa criar um portfólio de produtos e serviços inovador, maioritariamente baseado em cloud e tecnologias cognitivas, através de um forte esforço de investigação e desenvolvimento alinhado em torno de quatro fortes vetores tecnológicos de transformação: redes 5G, continuum de computação Edge/Cloud, tecnologias e modelos de negócios data-driven, e Inteligência Artificial.[11]

O projeto Power combina o desenvolvimento de modelos de previsão de sinais fisiológicos com tecnologias de tele-monitorização de forma a propor soluções para melhorar a gestão de pacientes com hipertensão ou diabetes. O objetivo científico é o uso de metodologias de inteligência computacional para o desenvolvimento de modelos personalizados, interpretáveis e dinâmicos na previsão de hipertensão ou desvios da glicémia das gamas normais e a sua utilização na recomendação de hábitos

de vida. Os modelos serão no contexto do dia a dia dos pacientes, permitindo fornecer-lhes informação contínua ao longo da vida.

Para o efeito será desenvolvida uma plataforma integrada de tele-monitorização, capaz de combinar as vantagens da monitorização remota permanente proporcionada por soluções móveis de baixo custo, com modelos de previsão e de recomendação, permitindo a implementação de estratégias de fomento de hábitos de vida saudável nestes doentes para além de facultar alarrmística inteligente aos profissionais de saúde.

1.2 Motivação

A presença de valores em falta e de anomalias são problemas recorrentes em qualquer conjunto de dados. Podem surgir devido ao mau funcionamento de um aparelho de medição ou devido à ausência de medições. Estes problemas podem provocar o mau funcionamento de certos algoritmos, levando a que sejam obtidos resultados e conclusões incorretos.

Atualmente já existem vários sistemas que fazem este tipo de tratamento de dados, no entanto estes permitem só usar um modelo já treinado, normalmente usando inteligência artificial. E são específicos só para situações de valores em falta ou deteção de anomalias.

Dada a existência de módulos de previsão e de recomendação para dados de pressão sanguínea e glicose, no projeto POWER, é essencial que os dados não contenham estes problemas, para que não se corra o risco de haver previsões e recomendações incorretas, o que nesta área poderá ser perigoso. Por isso, é essencial a criação de um módulo de pré-processamento de dados que detete e elimine anomalias e preencha valores em falta e que faça a seleção automática dos melhores algoritmos tendo em conta as variáveis usadas e o paciente de onde originam.

1.3 Objetivos

Tendo em conta o descrito na motivação, este projeto tem como objetivo a criação de um interface que seja capaz de:

- Deteção de anomalias

- Imputação de valores em falta.
- Fazer a seleção automática dos melhores algoritmos

Dada a necessidade de se precisar de dados que estejam completos e sem anomalias, para que possam ser usados pelas interfaces de previsão e de recomendação do sistema. É necessário criar uma interface que seja responsável por tratar do pré-processamento de dados. Isto é, que seja capaz de primeiro, detetar anomalias e por conseguinte eliminá-las. E segundo, que faça a imputação de valores em falta já existentes e que surjam da eliminação de anomalias. Por fim, deve ser capaz de selecionar os melhores algoritmos de forma automática, tendo em conta as variáveis presentes e o paciente de onde originam os dados. Tendo em conta a necessidade de determinar os melhores algoritmos, procura-se saber para os casos multivariáveis, se os algoritmos multivariáveis conseguem realmente obter melhores resultados no geral, em relação as algoritmos univariáveis.

1.4 Organização do documento

Esta dissertação está organizada em 6 capítulos.

No capítulo 1, é apresentado o enquadramento, motivação e objetivos para a realização da dissertação.

No capítulo 2, são revistos alguns conceitos do estado da arte relativos a algoritmos para imputação de valores em falta e para a deteção de anomalias.

No capítulo 3, é descrito a metodologia usada para a realização da interface. É mostrado os datasets, algoritmos, simulações e validações usados.

No capítulo 4, é exposta a interface realizada e a aplicação demonstrativa desta mesma interface, incluindo casos de uso e arquitetura.

No capítulo 5, são mostrados e discutidos os resultados obtidos da experimentação da interface.

No capítulo 6, é apresentado as conclusões obtidas e o trabalho futuro a ser desenvolvido.

Capítulo 2

Estado da Arte

2.1 Introdução

Este capítulo visa explicar os problemas da presença de anomalias e de valores em falta, nomeadamente, o que estes podem provocar numa análise de um determinado conjunto de dados e a necessidade de os resolver antes do processamento dos dados.

Será também e mostrado os principais tipos de algoritmos que são usados para a resolução destes problemas em específico.

2.2 Séries Temporais

Uma série temporal corresponde a um conjunto de observações de um processo, em que os intervalos entre observações são constantes, por exemplo semanas, meses, anos ou pequenos intervalos temporais [28]. As séries temporais distinguem-se dos outros tipos de dados longitudinais pelo número e fonte das observações. Um série temporal univariável contém muitas observações que originam de uma única fonte (por exemplo um indivíduo), enquanto que outros tipos de dados longitudinais podem conter dados de várias fontes (por exemplo um grupo de indivíduos. O tamanho da série temporal pode variar, mas geralmente contém pelo menos 20 medições e vários modelos requerem mais de 50 medições para se ter uma previsão precisa. Contudo mais dados que isto é sempre preferível, mas no mínimo a série temporal deve ter um comprimento que permita capturar o fenómeno de interesse.

As séries temporais, devido à suas estruturas únicas, exibem certas características: autocorrelação e estacionaridade. Também são representadas por quatro componentes: tendência, sazonalidade, ciclicidade e aleatoriedade.

2.2.1 Tendência

A tendência refere-se a qualquer mudança sistemática ao nível da série temporal, isto é, a sua direção a longo prazo [28]. Tanto a direção, como a inclinação da tendência pode manter-se constante ou mudar ao longo da série temporal. Isto pode ser visto como os valores da série temporal ficarem ao longo do tempo a maiores, menores ou estacionários.

Como a tendência representa uma fonte de variabilidade significativa, é aconselhado que seja tomada em conta quando se realiza análises sobre os dados da série temporal. Isto pode ser alcançado ao modelar explicitamente a tendência ou removendo-a usando métodos matemáticos. A primeira opção é usada quando a tendência é interessante, no entanto removê-la é feito quando não é uma componente interessante para os objetivos da análise.

2.2.2 Sazonalidade

A sazonalidade de um série temporal é um padrão de crescimento ou redução dos valores da série temporal que ocorre consistentemente durante a duração da série temporal. Ou seja, pode ser definido como um padrão cíclico ou repetitivo que ocorre num certo período, que normalmente está atribuído a fatores sazonais, isto é, fatores que estão associados a aspetos do calendário [28]. Pode-se pensar por exemplo, no caso do Algarve, em que o número de habitantes pode duplicar durante os meses do verão [29].

Esta característica, caso seja detetada, é muitas vezes removida por um procedimento conhecido como ajustamento sazonal. Este processo é realizado para simplificar os dados de forma a que estes sejam mais facilmente interpretados sem grande perda de informação. Pois a presença de sazonalidade pode mascarar a presença de outras características importantes, como por exemplo uma tendência ou um ciclo.

2.2.3 Ciclos

Um ciclo é um padrão na série temporal de aumento ou redução dos valores que se repete ao longo de períodos de tempo [28]. No entanto difere da sazonalidade pois não está associado a algum aspeto do calendário. O padrões representados por ciclos não têm uma duração fixa, isto é, a sua duração varia de ciclo para ciclo. Simplesmente, ciclos são componentes não sazonais que variam seguindo um padrão. São normalmente mais variáveis que os padrões sazonais.

2.2.4 Componente Irregular (Aleatoriedade)

A aleatoriedade ou componente irregular representa a série temporal residual após a remoção das outras componentes (tendência, sazonalidade e ciclicidade). Corresponde ao ruído estatístico e é análogo a erros presente em vários tipos de modelos estatísticos [28]. Quando esta componente é completamente aleatória, pode ser designada como ruído branco.

Esta componente surge de flutuações pequenas nos dados que não são sistemáticas e que em alguns casos não são previsíveis.

2.2.5 Autocorrelação

Autocorrelação é representada pela correlação de Pearson para uma variável consigo mesma num período de tempo passado, referido como o atraso da autocorrelação. Por exemplo, o atraso de ordem 1 da autocorrelação da série temporal corresponde à correlação de cada valor com a medição imediatamente anterior. Enquanto um atraso de ordem 2 é a correlação de cada valor com o valor que ocorreu a duas medições anteriores.

O poder da autocorrelação diminui consoante o atraso aumenta, dado que o valores utilizados ficam cada vez mais afastados.

2.2.6 Estacionaridade

Um problema comum presente em séries temporais corresponde ao facto de a média, variância e autocorrelação da série temporal variarem ao longo do tempo. Um série

temporal é considerada estacionária, caso estas propriedades mantenham-se constantes [28].

A estacionaridade é um conceito muito importante para a realização de análises sobre os dados da série temporal, porque as medidas estatísticas das séries temporais só são precisas caso se mantenham constantes ao longo da série temporal. Numa série temporal estacionário é mais fácil realizar a previsão de valores, pois estes manterão-se semelhante àqueles do passado. Por isso a estacionaridade da série temporal é a assunção mais importante quando se realiza previsões baseadas em valores passados e muitos modelos e algoritmos que usam séries temporais assumem que exista estacionaridade.

Contudo, muitas séries temporais estão dominadas por tendências e efeitos sazonais que impedem a estacionaridade. Como já foi referido, a tendência traduz numa alteração da série temporal ao longo do tempo, impedindo assim que o nível médio da série se mantenha ao longo do tempo. E efeito sazonais também, com as suas alterações que ocorrem em certos períodos do tempo alteram, a média dentro de um período fixo de tempo. Por isso estas duas componentes devem ser tratadas para garantir a estacionaridade.

2.3 Deteção de anomalias

Anomalias são valores extremos que se encontram afastados dos outros valores do dataset [3], ou valores que sejam inconsistentes com os restantes valores [7]. Estes podem indicar uma variação na medição, um erro experimental ou uma novidade. Ou seja, uma anomalia é uma observação que é diferente das restantes observações.

Existem três tipos de anomalias: pontuais, contextuais e coletivas [4]. Anomalias pontuais são instâncias individuais e isoladas que são consideradas anómalas em relação aos restantes pontos. São o tipo mais comum e simples e que atrai a maioria da atenção no que toca a investigação na deteção de anomalias. Anomalias contextuais são definidos como instâncias cujo seu comportamento se desvia de outras instâncias com informação contextual semelhante [12]. Anomalias coletivas são grupos de instâncias cujo o seu comportamento, num todo, é anómalo, isto é, estas instâncias sozinhas podem não ser consideradas anomalias, mas, em conjunto, o seu comportamento é anómalo em relação ao resto do dataset.

Anomalias podem ser provocadas por diferentes causas, sendo as mais comuns as seguintes:

- Erros de introdução de dados (erros humanos);
- Erros de medição (erros instrumentais);
- Erros experimentais (erros de extração de dados ou de planeamento/execução da experiência);
- Erros de processamento de dados
- Erros de amostragem (erros de extração ou mistura de dados obtidos de diferentes fontes)
- Ocorrências naturais (novidades nos dados, não são considerados erros)

A presença de anomalias podem provocar alterações à distribuição de dados que por sua vez provoca a alteração de certas medidas estatísticas. Isto é, a presença de outliers elevados e/ou baixos levam à alteração da média e do desvio padrão de um certa distribuição de dados. Provocando a possível ocorrência de erros, como por exemplo em inferências estatísticas[5], dado que estas são baseadas em testes de médias em que o desvio padrão é usado como uma medida da flutuação normal dos processos examinados, originando erros do tipo-I e do tipo -II. Mais especificamente, também podem provocar consequências negativas em algoritmos que lidam com séries temporais [14], levando a erros de especificação no modelo, estimação de parâmetros enviesada e más previsões.

Tendo em conta a consequências mencionadas, a deteção de anomalias é um processo essencial para a maioria das áreas quantitativas (Física, Economia, Finanças, Machine Learning, Cíber Segurança), visto que a qualidade dos dados é tão importante quanto a qualidade da previsão ou do modelo de classificação [7].

2.3.1 Algoritmos estatísticos

Algoritmos estatísticos para a deteção de anomalias baseiam-se na distribuição dos dados [7]. Logo, estes métodos são otimizados para cada distribuição dependente em

parâmetros específicos da distribuição correspondente, o número esperado de anomalias e a gama de valores onde são esperados ser encontrados.

Uma regra muito usada e simples é considerar que cada valor que esteja uma distância da média de uma distribuição normal de três vezes o desvio padrão pode ser considerado uma anomalia.

Um problema que existe com este tipo de algoritmos é o facto de se precisar de assumir uma determinada distribuição para se aplicar um determinado teste, por exemplo em muitos testes estatísticos é necessário ter-se uma distribuição normal dos dados. Quer para métodos univariáveis, quer para métodos multivariáveis, é necessário ter-se uma determinada distribuição para os aplicar. Uma potencial solução é a ajustar uma distribuição gaussiana ao dataset, ou outra solução é usar a distância de Mahalanobis, baseada na matriz de covariância Σ como uma medida de anormalidade.

Contudo, para realizar estes testes é necessário usar a média, o desvio padrão ou a covariância. No entanto estas medidas em si são sensíveis a outliers e quando se faz a computação destes valores é necessário usar os outliers presentes no dataset. Logo isto pode levar a um processo conhecido como "masking" [7], em que os outliers escondem a sua própria presença ao influenciarem o cálculo dos parâmetros a serem utilizados nos algoritmos, originando possivelmente falsos positivos.

Este tipo de algoritmos é considerado para este estudo, dado que são algoritmos clássicos e muito conhecidos, e na sua maioria bastante simples e que requerem pouco poder de processamento. Podendo-se evitar assim o uso de algoritmos mais complexos caso este consigam obter bons resultados. São normalmente usados para situações univariáveis, que é um dos alvos de estudo

2.3.2 Algoritmos de clustering

Algoritmos de clustering são importantes para lidar com datasets de grande dimensões [6]. Estes algoritmos procuram grupos homogéneos no dataset formando assim clusters que depois podem ser analisados separadamente. Este tipo de algoritmos, na sua generalidade, procuram primeiro identificar grupos de pontos que sejam semelhantes e numa segunda fase, esta usada para a deteção de anomalias,

identificam os pontos que tenham ficado fora de clusters ou que estejam muitos distanciados do centro do seu cluster como anomalias [16].

O k-means é uns do algoritmos mais conhecidos, usado especialmente para clusters esféricos [6], em que selecionam k centros e posteriormente para cada ponto procura-se o centro cuja distância seja mínima e assim associando esse ponto a esse cluster, contudo não é um método robusto pois usa as médias do grupo para selecionar o centro. Outro método que procura negar este problem é o k-medoids, visto que este usa como ponto central do cluster um ponto existente no dataset, sendo o passos seguintes semelhantes ao k-means.

Um problema com uma grande parte destes algoritmos é o facto de precisarem que o analisador introduza valores de entrada, como por exemplo o número de clusters a serem formados (k-means, k-medoids) ou a distância entre os pontos (DBSCAN). Por isso, é necessário algum cuidado na seleção destes parâmetros para se obter os melhores resultados.

Este tipo de algoritmos são também muito conhecidos e já bastante testados para diferentes tipos de situações, sendo uma delas a deteção de anomalias. Por isso são utilizados neste estudo.

2.4 Imputação de dados em falta

Valores em falta são definidos como vamos valores que não foram guardados para uma determinada instância no dataset, podendo ser só uma variável em falta ou várias [2]. é um problema que está presente numa grande parte de trabalhos de investigação e que pode afetar os resultados e conclusões obtidos.

Existem três tipos de dados em falta, tendo por base a razão da sua ausência [2,17]. O primeiro é o MCAR, que representa os valores em falta cuja sua probabilidade não está relacionada com o valor em específico que é suposto obter ou o conjunto de dados observados. Este tipo de dados em falta pode surgir devido a falhas de equipamento de medição, perdas de amostras durante o seu transporte ou porque os dados não satisfatórios. Dados em falta do tipo MAR são representados por dados cuja sua probabilidade de estar em falta depende de instâncias observadas, mas não estão relacionados com valores em específico que se esperam obter. É o tipo mais

comum e abrangente de dados em falta, cujo a probabilidade de os dados estarem em falta depende de uma causa conhecida. MCAR acontece quando nenhum dos outros tipos se verifica. São considerados casos complicados, cuja a probabilidade dos dados estarem em falta é provocada por algo fora do nosso conhecimento.

A presença de valores em falta pode originar alguns problemas[2]. O poder estatístico pode ser reduzido, provocando a o aumento da probabilidade da hipótese nula ser rejeitada quando é verdadeira, pode enviesar a estimação de parâmetros, pode reduzir a representação das amostras e, por fim, pode complicar a análise realizada para uma determinada investigação. Cada um destes problemas pode ameaçar a validade de um estudo e pode causar conclusões inválidas. Assim para resolver este problema é necessário realizar o processo de imputação, que corresponde ao processo de preencher valores em falta por valores aceitáveis [9].

2.4.1 Imputação Simples

Algoritmos de imputação simples correspondem a processos de imputação em que substituí cada valor em falta uma única vez [8]. Os valores em falta podem ser substituídos por um único valor que represente melhor o mecanismo dos valores em falta [10]. Podendo-se utilizar a média de uma variável que seja contínua e que tenha uma distribuição normal, a mediana ou moda de uma variável categórica, o valor anterior ou posterior ao valor em falta ou um valor calculado a partir de uma regressão.

Este método é aceitável para situações em que a proporção de valores em falta seja pequena (pro exemplo, menos de 5%) [9] e proporciona uma solução simples e rápida. Todavia esta solução, sem medidas corretivas, tende a exagerar a precisão, dado que omite a componente de variabilidade entre imputações. Também não é eficaz para situações em que existam várias variáveis que se relacionam entre si. Considere-se o seguinte exemplo em que substituímos os valores em falta pela média da variável. O que acaba por acontecer é diminuição da variabilidade, visto que o mesmo valor é sempre introduzido. E não se consideram outras variáveis para a imputação do valor, perdendo assim as possíveis relações que existiam.

Estes algoritmos correspondem ao tipo mais simples de imputação e foram escolhidos para este estudo devido à necessidade de experimentar métodos simples e

rápidos para o caso de poderem ser mais eficazes para algumas situações e assim evitar o uso de algoritmos mais complexos.

2.4.2 Imputação Múltipla

Imputação múltipla é uma técnica de Monte Carlo, que cria múltiplos datasets imputados, cada um com diferentes valores imputados [8, 9] e que tenta ultrapassar as desvantagens da imputação simples [10].

Este método é composto por três fases: imputação, análise e junção dos resultados [10]. Na primeira fase várias cópias do dataset incompleto, que depois são preenchidas por valores que são escolhidos aleatoriamente de distribuições estimadas a partir dos dados observados, com condicionalidade em relação às outras variáveis [8, 10]. Este processo introduz variância aos valores imputados. Na fase de análise, cada dataset, agora completos, são analisados usando métodos padrão [9]. Na fase final, os resultados são agrupados ao fazer-se a média das estimativas de cada análise para derivar a estimativa da junção e intervalos de confiança que incorporam a incerteza dos valores em falta.

Ao contrário da imputação simples, a imputação múltipla oferece resultados válidos quando os dados em falta são do tipo MAR. Isto deve-se às diferenças sistemáticas entre valores em falta e valores observados, que são originadas pela informação já presente nos dados que são possuídos, e isto é um fator que é considerado na distribuições preditivas utilizadas na primeira fase da imputação múltipla [10].

Estes algoritmos foram selecionados para serem utilizados devido ao facto de serem usados especificamente para a imputação de valores em falta em casos multivariáveis, que um dos alvos de estudo deste projeto.

Capítulo 3

Métodos

3.1 Introdução

Este capítulo visa explicar o procedimento utilizado para a realização da experimentação do módulo de pré-processamento de dados. Inclui a descrição dos datasets utilizados. Depois, é feita uma descrição de cada uma das componentes do módulo, correspondentes aos objetivos apresentados, cada uma contendo a descrição dos algoritmos utilizados, o processo de simulação do problema que foi utilizado e por fim informação relativa à validação que foi utilizada para comparar cada algoritmo.

3.2 Bases de dados utilizadas

3.2.1 myHeart

A base de dados myHeart contém dados de pacientes em que foram medidos os valores de pressão sanguínea, mais precisamente os valores registados correspondem à pressão sanguínea sistólica. Esta é a variável considerada principal para este estudo. Outras variáveis foram utilizadas em combinação com esta, sendo elas o ritmo cardíaco, peso e ritmo respiratório. Para este estudo considerou-se uma situação univariável só com a pressão sanguínea e subsequentemente criaram-se outros datasets multivariáveis utilizando combinações das variáveis secundárias com a principal, por exemplo pressão sanguínea com ritmo cardíaco ou a junção de todas as variáveis.

O dataset completo incluí dados de vários pacientes e algumas instâncias encontram-se incompletas ou ausentes. Logo houve a necessidade de fazer um tratamento prévio em que se procurou a maior sequência de dias consecutivos, obtendo um paciente com quarenta e cinco dias registados consecutivamente, sem valores em falta, sendo este o utilizado.

3.2.2 Dataset da APDP

A base de dados da APDP contém dados de pacientes em que foram medidos os valores de glicose no sangue. Neste caso a variável principal é a quantidade de glicose no sangue do paciente, medida em mg/dL. Outras variáveis não foram consideradas, porque não tinham interesse para esta experiência. Logo neste caso considera-se somente o caso univariável com a variável principal.

O mesmo processo de tratamento prévio de dados, usado para o dataset myHeart, foi utilizado para este dataset. As mesmas razões já mencionadas anteriormente servem para justificar este passo da experiência.

Usou-se os dados de um paciente, onde obteve-se uma sequência de mais de 2000 valores de glicose consecutivos, cada um com um intervalo de cinco minutos.

3.3 Algoritmos para deteção de outliers

3.3.1 Descrição

3.3.1.1 Intervalo Interquartil

Tal como a regra dos três desvios padrões, em que se considera que qualquer valor que esteja a uma distância de três desvios padrões da média da distribuição normal é uma anomalia, neste método também se calcula uma distância para determinar que valores são outliers, no entanto pode ser utilizado para outras distribuições de dados que não sejam distribuições normais.

Para determinar a distância a usar para fazer a deteção de anomalias, é necessário calcular os quartis dos dados do dataset, ou seja, primeiro quartil ou quartil inferior (Q1) que representa 25% dos dados e o terceiro quartil ou quartil superior (Q3) que

representa 75% dos dados [24]. O segundo quartil ou mediana (Q2) não é necessário. Depois, é necessário calcular o intervalo interquartil (IQR) ao fazer a subtração entre Q3 e Q2. Para o cálculo destes valores é necessário que os dados estejam ordenados.

As anomalias são os valores que estejam abaixo de Q1 ou acima de Q3 uma distância de $1{,}5 \times IQR$.

O facto deste algoritmo ser bastante simples e permitir que seja utilizado em qualquer tipo de distribuição, fez com que este algoritmo fosse selecionado.

3.3.1.2 DBSCAN

DBSCAN é um algoritmo de clustering espacial baseado em densidade, que pode detetar anomalias [23]. Para usar este algoritmo é necessário definir os valores de duas variáveis de entrada: a distância da vizinhança epsilon (eps) e o número mínimo de pontos da vizinhança (minpts). O algoritmo começa por selecionar arbitrariamente um ponto do dataset e procura todos os pontos que estejam à distância eps do ponto selecionado, sendo esses pontos considerados seu vizinhos. Caso o número de pontos encontrados seja maior ou igual a minpts o algoritmo considera que o ponto selecionado é um ponto central, logo pode ser selecionado para formar um cluster [15, 23]. Após todos o pontos terem sido percorridos e classificados, o algoritmo seleciona um dos pontos centrais e cria o primeiro cluster e adiciona todos os outros pontos centrais que estejam na vizinhança que estejam na vizinhança desse ponto. Em seguida percorre iterativamente os outros pontos que foram adicionados ao cluster e efetua o mesmo processo de adicionar mais pontos centrais ao cluster. Quando não encontra mais pontos centrais na vizinhança desse cluster, procede ao adicionamento de outros pontos que não foram classificados como centrais, mas que tenham como vizinho pelo menos um ponto central. Note-se que com estes pontos não se efetua o processo de adicionar os pontos da sua vizinhança ao cluster, caso ainda não pertençam. Estes processos são repetidos até que não seja possível formar mais clusters.

No DBSCAN os pontos podem ser classificados como pontos centrais, pontos fronteira e pontos anómalos. Os pontos centrais, já definidos anteriormente, são os pontos que são selecionados e qua obedecem aos parâmetros de entrada para formarem um cluster. Os pontos de fronteira são pontos que são vizinhos de pelo

menos um ponto central, mas que não obedecem aos parâmetros de entrada. O pontos anómalos são pontos que não têm como vizinho um ponto central e que não obedecem aos parâmetros de entrada.

Para o processo de deteção de anomalias não é necessário efetuar o processo de formação de clusters descrito anteriormente. É somente necessário efetua a classificação dos pontos para saber que pontos são anomalias. O algoritmo descrito abaixo demonstra o funcionamento do algoritmo.

```
Inputs:
D: the dataset
Eps: the neighborhood distance
Minpts: the minimum number of points
Output:
Discovered outliers and clusters
Variables:
m, n: row and column values of D matrix, respectively
Dist: distance vector
indices: indices that distance of points is lower than Eps
class_no: indicates the clusters – default 1

Algorithm:
1.  import the data-set into D
2.  for i = 1 to m   //row counter
3.    Dist = distance(i, D)
4.    neighbors= find(Dist =< Eps)
5.    neighbor_count = count(neighbors)
6.    core_neig=check_core_neighbor (neighbors)
7.    if (neighbor_count >=minpts)
8.      class(i) = class_no   //clustered point
9.      while(more points near i)
10.       class(point) = class_no
11.     end while
12.     class_no += 1
13.   else if(neighbor_count<minpts & core_neig==True)
14.     class(i) = 0          //border point
15.   else if (neighbor_count<minpts)
16.     class(i) = -1         //outlier point
17.   end if
18. end for
19. return class
```

Figura 3.1: Algoritmo representativo do funcionamento do DBSCAN para deteção de anomalias [23]

O algoritmo foi escolhido devido ao facto de ser um algoritmo bastante conhecido, com vários artigos em foi utilizado para esta situação da deteção de valores em falta. A sua escolha também deve-se ao facto de no seu funcionamento normal permitir detetar anomalias sem a adição de métodos extra ao algoritmo original, algo que não é possível noutros algoritmos de clustering.

3.3.1.3 Isolation Forest

O algoritmo de deteção de anomalias, denominado Isolation Forest (iForest), consiste num processo dividido em duas fases [26]. A primeira fase corresponde à fase de treino em que são construídas isolation trees (iTrees) usando amostras do dataset de treino. Na segunda fase, também nomeada fase de teste, passa as instâncias do dataset de teste pelas isolation trees para obter um pontuação que identifica a instância como anómala ou não.

Na fase de treino, as isolation trees são construídas ao particionar recursivamente o dataset usado para treino do modelo. Ou seja, a formação da iTree começa ao selecionar uma variável aleatoriamente e usa-se um limite aleatório, que corresponde a um valor entre os valores máximo e mínimo da variável. Se o ponto fôr inferior ao limite fica no ramo da esquerda, se fôr maior fica na direita. Este processo repete-se até que cada instância esteja isolada ou que seja alcançada uma altura da árvore pré-definida. Esta fase contém duas variáveis de entrada. A primeira é o número de iTrees a serem criadas (t) e a segunda é o tamanho da amostras (ψ), sendo que esta última controla o tamanho da dataset de treino. Os algoritmos das figuras 3.2 e 3.3 demonstram o funcionamento desta fase de treino.

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - sub-sampling size

Output: a set of t *iTrees*

1: **Initialize** $Forest$
2: set height limit $l = ceiling(\log_2 \psi)$
3: **for** $i = 1$ to t **do**
4: $X' \leftarrow sample(X, \psi)$
5: $Forest \leftarrow Forest \cup iTree(X', 0, l)$
6: **end for**
7: **return** $Forest$

Figura 3.2: Algoritmo representativo do funcionamento do Isolation Forest durante a fase de treino [26]

Algorithm 2 : $iTree(X, e, l)$

Inputs: X - input data, e - current tree height, l - height limit

Output: an iTree

1: **if** $e \geq l$ or $|X| \leq 1$ **then**
2: return $exNode\{Size \leftarrow |X|\}$
3: **else**
4: let Q be a list of attributes in X
5: randomly select an attribute $q \in Q$
6: randomly select a split point p from max and min values of attribute q in X
7: $X_l \leftarrow filter(X, q < p)$
8: $X_r \leftarrow filter(X, q \geq p)$
9: return $inNode\{Left \leftarrow iTree(X_l, e+1, l),$
10: $Right \leftarrow iTree(X_r, e+1, l),$
11: $SplitAtt \leftarrow q,$
12: $SplitValue \leftarrow p\}$
13: **end if**

Figura 3.3: Algoritmo da criação de uma iTree [26]

Na fase de teste, é determinada a pontuação para determinar se uma instância é uma anomalia ou não, ao determinar o tamanho expectável do caminho percorrido pela instância. Este valor é alcançado ao passar cada instância por cada iTree da iForest. Usando o algoritmo 3, o tamanho do caminho é calculado ao contar o número de arestas que a instância percorre dentro da iTree, desde da raiz até à folha alcançada pela instância. Após percorrer todas as iTrees é feita uma agregação de todos os valores de profundidade obtidos e usando a equação 1, em que $E(h(x))$ corresponde à média dos caminhos obtidos, $c(n)$ é média de dos caminhos obtidos usando n, que corresponde à altura máxima possível para uma iTree. É calculada a pontuação $s(x, n)$.

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \quad (1)$$

O valor de s(x, n) determina se a instância é anomalia ou não:

- Se s(x, n) estiver muito próximo de 1, então é muito provável que o valor seja uma anomalia.
- Se s(x, n) é menor que 0,5 então é um valor considerado normal
- Se todas a instâncias retornarem s(x, n) próximo de 0,5, então a amostra não tem qualquer anomalia.

Algorithm 3 : $PathLength(x, T, e)$

Inputs : x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called

Output: path length of x

1: **if** T is an external node **then**
2: return $e + c(T.size)$ $\{c(.)$ is defined in Equation 1$\}$
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: return $PathLength(x, T.left, e + 1)$
7: **else** $\{x_a \geq T.splitValue\}$
8: return $PathLength(x, T.right, e + 1)$
9: **end if**

Figura 3.4: Algoritmo do processo do cálculo do tamanho de caminho para cada valor. [26]

O algoritmo foi selecionado devido ao facto de também ser um algoritmo muito conhecido e que permite que seja utilizado com diferentes tipos de dados (por exemplo numéricos e categóricos), algo que não está presente neste estudo, mas que pode vir a estar presente no futuro.

3.3.2 Abordagem para simular anomalias

A técnica usada para esta parte do estudo foi criada ad-hoc, para fazer a criação de anomalias no dataset. Primeiro são selecionados índices aleatoriamente, correspondentes a valores do dataset, para serem anomalias. Em segundo, os valores que foram selecionados são incrementados ou reduzidos por um valor correspondente a uma certa percentagem do valor original (10, 20, 30%), criando assim outliers com diferentes distâncias dos restantes valores. Isto é, cria-se assim situações em que é mais difícil distinguir anomalias dos restantes valores e outras em que são mais facilmente detetados. A percentagem de anomalias também é modificável, sendo que neste caso usou-se 5, 10, 15% de anomalias

3.3.3 Avaliação

A avaliação e consequente comparação dos algoritmos foi realizada utilizando uma matriz com o resultado dos algoritmos, esta contendo os índices do valores que foram previstos como anomalias, e a matriz inicial que continha os índices dos valores que eram anomalias, correspondendo às anomalias verdadeiras. Com estes dois

conjuntos de valores são determinadas as quantidades de falsos positivos, falso negativos, positivos verdadeiros e negativos verdadeiros, usando a matriz de confusão. Em seguida são calculados os valores de detection rate e false alarm rate, que são calculados usando cada uma combinação dos valores anteriores (equações 2 e 3). E posteriormente são determinadas as curvas ROC para cada algoritmo que representam o trade-off entre este dois valores. Como estas curvas são representadas num gráfico 2-D é possível determinar a AUC que corresponde à área abaixo da curva que é utilizada como medida de avaliação dos algoritmos [30].

$$\textit{Detection rate} = \mathrm{TP} / (\mathrm{TP} + \mathrm{FN}) \qquad (2)$$

$$\textit{False alarm rate} = \mathrm{FP} / (\mathrm{FP} + \mathrm{TN}) \qquad (3)$$

3.4 Algoritmos para imputação de dados em falta

3.4.1 Descrição

3.4.1.1 Imputação simples com média

Neste método os valores em falta são substituídos uma única vez. Neste caso usa-se a média da variável a que o valor em falta corresponde. Ou seja, itera-se sobre cada variável do dataset e calcula-se a média usando os valores correspondentes a essa variável que estejam presentes. Em seguida os valores em falta que existam nessa variável são imputados com a média calculada anteriormente.

3.4.1.2 MICE

O MICE é um algoritmo de imputação múltipla, que assume que os valores da variáveis encontram-se numa situação de MAR [18]. Este algoritmo não é eficaz para situações em que o dados em falta sejam MNAR, pois pode levar a resultados enviesados.

O algoritmo contém quatro passo principais. No primeiro passo é efetuada uma imputação simples, por exemplo usando a média de cada variável, para preencher os

seus valores em falta. Sendo que estes são valores temporário que serão somente usados para que as fases seguintes possam ocorrer.

No seguinte passo, uma das variáveis é selecionada e todos os seus valores em falta que tinham sido imputados voltam a ficar omitidos.

No terceiro passo, é efetuada a regressão usando os valores existentes da variável escolhida com os valores das outras variáveis. Ou seja, a variável escolhida é considerada a variável dependente da regressão, enquanto que as restantes são consideradas variáveis independentes. Pode-se utilizar vários modelos de regressão, no entanto o escolhido para este estudo foi a regressão linear.

Na fase final, os valores em falta da variável escolhida são substituídos com previsões do modelo de regressão. Os passos dois a quatro são depois repetidos para as restantes variáveis. Os valores imputados no quarto passo são utilizados para fazer a regressões das outras variáveis.

O conjunto destes passos constituí uma iteração do algoritmo, como foi mencionado no ponto 2.3.2, os algoritmos de imputação múltipla fazem várias iterações para o mesmo valor em falta antes de alcançar um resultado final. A cada iteração, os valores imputados são atualizados. Estas iterações são repetidas até alcançar-se uma convergência ou até o número de iterações ser alcançado.

3.4.1.3 K-Nearest Neighbours

O algoritmo KNN para imputação de valores em falta é constituído por três fases [19]. É focado em utilizar os valores similares ao caso que se pretende imputar.

Na primeira fase é feita a seleção do valor de k, que corresponde ao número de vizinhos do ponto que queremos imputar. Este valor pode ser tomar ser qualquer número natural.

No segundo passo, são calculadas as distâncias entre o valor em falta e os seus k vizinhos mais próximos. Pode-se utilizar diferentes medidas de distâncias, por exemplo distância euclidiana ou distância Manhattan. É importante lembrar que este valores são guardados em memória, e quanto maior o valor de vizinhos a considerar (k), maior será a quantidade de memória utilizada. Neste estudo foi utilizada a distância euclidiana.

Por fim, os valores com distância mínima são escolhidos. E procede-se ao cálculo da média destes valores e é efetuada a imputação deste valor, no valor que se encontrava em falta.

Com este algoritmo é possível efetuar a imputação de valores discretos e de valores contínuos, sendo que no caso de valores discretos é usada a moda dos valores mais próximos e no caso contínuo usa-se a média. No entanto este algoritmo tem um problema de que para cada valor que se precisa de imputar tem-se de percorrer o dataset todo à procura dos vizinhos mais próximos, o que pode provocar algumas dificuldade de processamento em datasets muito grandes.

Este algoritmo foi selecionado devido ao facto de ser também simples e permitir que se usem diferentes tipos de valores. E também é um algoritmo com vários artigos criados sobre o seu uso em diferentes situações.

3.4.1.4 MissForest

O algoritmo MissForest é um algoritmo iterativo que utiliza como base o algoritmo Random Forest para realizar imputação de valores em falta [21].

O algoritmo Random Forests é capaz de fazer classificação e regressão [27], sendo que o utilizado para imputação de valores em falta é a sua capacidade de fazer regressão. Neste algoritmo são criadas múltiplas árvores, usando para cada uma delas uma amostra do dataset de treino. Após a criação da floresta, é passado cada ponto com valores em falta por cada árvore até se alcançar um nó folha obtendo-se assim um resultado para ser imputado. Quando todas as árvores forem percorridas, são agregados todos os valores obtidos e é feita a média, obtendo-se no final o valor pretendido.

No algoritmo missForest é feito para cada coluna ou variável a criação de quatro variáveis:

- O valores observados da variável s são denotados como $y^{(s)}_{obs}$,
- Os valores em falta da variável s são denotados como $y^{(s)}_{miss}$,

- Os valores observados nas outras variáveis referentes cujos índices coincidem com os valores da variável s que não têm valores em falta, denotados como $x^{(s)}_{obs}$,
- Os valores das outras variáveis cujos índices correspondem aos valores em falta da variável s, denotados como $x^{(s)}_{miss}$.

Note-se que no início do algoritmo, o valores em falta são imputados com a média ou moda da variável em que estão presentes.

Cada variável por ordem crescente da percentagem de valores em falta que possui, sendo que a primeira variável a ser imputada é aquela que possui um número menor de valores em falta.

No passo seguinte é feita a criação de uma random forest para a variável s usando os valores de $y^{(s)}_{obs}$ e $x^{(s)}_{obs}$. Em seguida é feita a previsão dos valores em $y^{(s)}_{miss}$ usando a random forest criada e os valores de $x^{(s)}_{miss}$.

Após todas as variáveis serem imputadas é feita a comparação do novo dataset imputado com o anterior, fazendo a diferença entre as imputações de ambos os datasets. Caso a diferença seja significativa ao ponto de alcançar o critério γ, o algoritmo acaba e retorna o dataset completo. Em caso negativo é feita uma nova iteração das imputações. O algoritmo 1 demonstra o processo descrito.

Algorithm 1 Impute missing values with RF.

Require: $\mathbf{X}$ an $n \times p$ matrix, stopping criterion γ

1. Make initial guess for missing values;
2. $\mathbf{k} \leftarrow$ vector of sorted indices of columns in $\mathbf{X}$ w.r.t. increasing amount of missing values;
3. **while** not γ **do**
4. $\quad \mathbf{X}^{imp}_{old} \leftarrow$ store previously imputed matrix;
5. $\quad$ **for** s in $\mathbf{k}$ **do**
6. $\quad\quad$ Fit a random forest: $\mathbf{y}^{(s)}_{obs} \sim \mathbf{x}^{(s)}_{obs}$;
7. $\quad\quad$ Predict $\mathbf{y}^{(s)}_{mis}$ using $\mathbf{x}^{(s)}_{mis}$;
8. $\quad\quad \mathbf{X}^{imp}_{new} \leftarrow$ update imputed matrix, using predicted $\mathbf{y}^{(s)}_{mis}$;
9. $\quad$ **end for**
10. $\quad$ update γ.
11. **end while**
12. **return** the imputed matrix $\mathbf{X}^{imp}$

Figura 3.5: Algoritmo de imputação usando Random Forest [21]

O algoritmo foi escolhido devido ao facto de ser um algoritmo um bocado mais complexo, que por sua vez também permite a utilização de dados diferentes tipos de dados. Foi também escolhido pelo facto de utilizar árvores de decisão, pois com este estudo pretende-se explorar diferentes tipos de algoritmos.

3.4.1.5 VAR-IM

O VAR-IM é um algoritmo autorregressivo que usa vetores para imputação de valores em falta em séries temporais multivariáveis [20]. O algoritmo usa uma combinação entre o algoritmo VAR, algoritmo EM e o algoritmo PEM para realizar a sua função. O algoritmo é considerado autorregressivo devido ao facto de usar, para a previsão de um determinado ponto, os valores outros pontos que antecedem o valor que se pretende determinar. É considerado vetorial visto que para a previsão de um ponto é feito o cálculo de um conjunto de equações cada uma para uma variável desse ponto, formando assim um vetor de equações ou um vetor de valores a serem calculados. Considere-se uma situação de um dataset com duas variáveis e um VAR com atraso de ordem 1. As equações deste VAR estão representadas pelas equações 4 e 5, onde c é uma constante, $y_{1,t}$ e $y_{2,t}$ representam os valores que se pretendem prever. $e_{1,t}$ e $e_{2,t}$ representam ruídos brancos que podem estar correlacionados. E as restantes variáveis representam o coeficientes que capturam a influência do atraso de uma variável consigo mesma (11,1 e 22,1) e a influência do atrasado de uma variável sobre a outra (12,1, variável 2 sobre variável 1 e 21,1 variável 1 sobre variável 2).

$$y_{1,t} = c_1 + \phi_{11,1} y_{1,t-1} + \phi_{12,1} y_{2,t-1} + e_{1,t} \quad (4)$$

$$y_{2,t} = c_2 + \phi_{21,1} y_{1,t-1} + \phi_{22,1} y_{2,t-1} + e_{2,t}, \quad (5)$$

Para iniciar, o algoritmo necessita de um dataset completo, devido ao facto de ser autorregressivo e por isso necessitar de valores passados para prever um determinado valor.. Logo é necessário imputar os valores em falta existentes. O algoritmo aconselhado e utilizado em [20] foi a interpolação linear.

O próximo passo é selecionar o melhor valor de atraso (p). Este valor corresponde a número de valores passados que afetam a previsão de uma valor. Por exemplo, com um p igual a um, para realizar a previsão de um valor, seria só considerado o valor anterior para contribuir para o cálculo da previsão. O melhor valor de p pode ser

calculado usando diferentes métricas. Neste caso foi utilizado o AIC para determinar o valor de p, isto é o valor de p que conseguir um valor mínimo no AIC é o escolhido.

Em seguida, procede ao uso de EM, isto é, o modelo é treinado e depois são utilizados os valores previsto para preencher o dataset incompleto (expectation). Depois volta-se a treinar um novo modelo VAR utilizando o dataset atualizado e comparam-se o valores das matrizes de covariância (maximization), se houver convergência entre as duas o algoritmo avança para o passo seguinte, em caso negativo volta a repetir o processo de criar um novo modelo e de comparar com o anterior.

No fim, o algoritmo usa o PEM para reduzir o seu erro em comparação aos dados originais, obtendo um modelo final a ser utilizado. O esquema seguinte mostra a sequência de ações do algoritmo descrito.

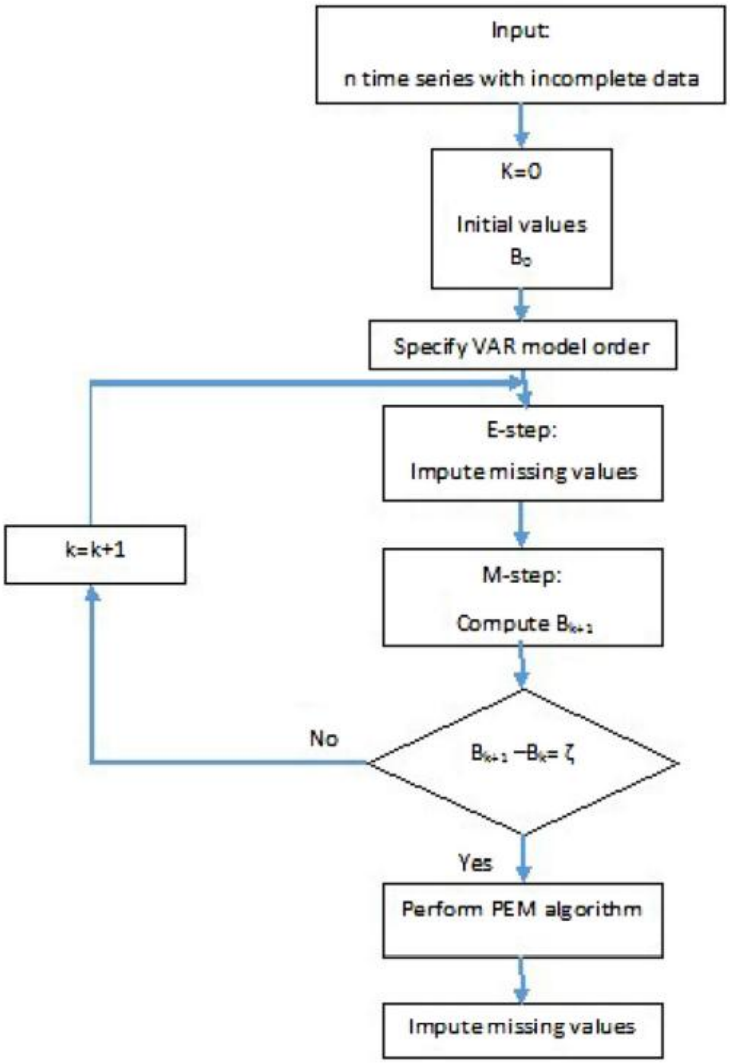

Figura 3.6: Esquema representativo do funcionamento do VAR-IM [20]

Este algoritmo foi escolhido, porque é um algoritmo autorregressivo, isto é um algoritmo criado especificamente para fazer a imputação de valores em falta em séries temporais. Para além disso é utilizado para séries temporais multivariáveis, que são alvo de estudo deste projeto. No artigo em que este algoritmo é explicado, este foi usado para a imputação de valores em falta em séries temporais com dados clínicos, que por sua vez corresponde ao tipo de dados usados neste projeto.

3.4.1.6 ARIMA

O modelo ARIMA é utilizado para resolver problemas de previsão em séries temporais univariáveis [22], e é considerado um algoritmo autorregressivo. O seu uso foca-se maioritariamente no previsão de valores futuros, ou seja,valores além dos

que se encontram no dataset, no entanto também possível treinar o modelo para fazer previsões dentro da amostra usada para treinar o modelo. Isto permite que seja possível realizar previsões de valores em falta.

O modelo contém três variáveis de entrada. Sendo estas: a ordem de autorregressão (p), o grau de diferenciação (d), e a ordem da componente móvel média (q). O modelo pode ser representado pela equação 6, onde L é o operador de retrocesso.

$$\left(1-\sum_{i=1}^{p} a_i L^i\right)(1-L)^d y(t) = \left(1+\sum_{i=1}^{q} \beta_i L^i\right) y(t)\xi(t) \qquad (6)$$

Para realizar previsões de valores em falta, o modelo primeiro treina com dataset completo, por isso é necessário fazer uma imputação com, por exemplo, interpolação. Depois de treinado, procede a realizar as previsões dos valores em falta, um de cada vez. Caso haja valores em falta consecutivos, é necessário usar valores previsto previamente para realizar a previsão.

O algoritmo foi escolhido devido a ser um algoritmo de autorregressão muito conhecido para situações univariáveis.

3.4.2 Abordagem para simular dados em falta

A abordagem adotada para a simulação de valores em falta é composta por quatro fases [25].

Na primeira fase o dataset é dividido em secções aleatórias, com número igual ao número de padrões que se pretendem introduzir. Um padrão é definido pela percentagem de pontos/linhas do dataset que é suposto influenciar, pelo tipo de valores em falta que se pretende usar (MCAR, MAR ou MNAR), pelas variáveis que são suposto terem valores em falta, pelos pesos que cada variável tem no processo que decide a probabilidade de um valor dessa variável ficar omitido e por uma distribuição (left-tailed, right-tailed, centrada ou both-tailed), representadas na figura 3.7. Por exemplo numa distribuição left-tailed, as variáveis que tenham pesos menores são as que têm maior probabilidade de ter valores em falta. Contudo esta componente dos pesos só é considerada para situação de MNAR. Numa situação de MAR ou MCAR o peso é sempre zero, ou seja todas as variáveis do padrão iram ter a

mesma probabilidade dos seu valores serem omitidos. A soma de todas a percentagens de pontos/linhas do dataset de cada padrão deve ser igual a 1.

Na segunda fase é feita a soma dos pesos que irão determinar a probabilidade de uma determinada célula do dataset ficar em falta.

Na terceira fase são determinados os valores que ficam em falta e aqueles que se mantêm, tendo em conta a probabilidade calculada na fase anterior.

Por fim, as secções voltam a ser agrupadas pela ordem original, retornando o mesmo dataset, mas com valores em falta.

O funcionamento deste algoritmo pode ser visualizada através do esquema

Escolheu-se este método para a criação de valores em falta, pois desta maneira evita-se usar um método criado ad-hoc. E com este método pretende-se usar um método estandardizado.

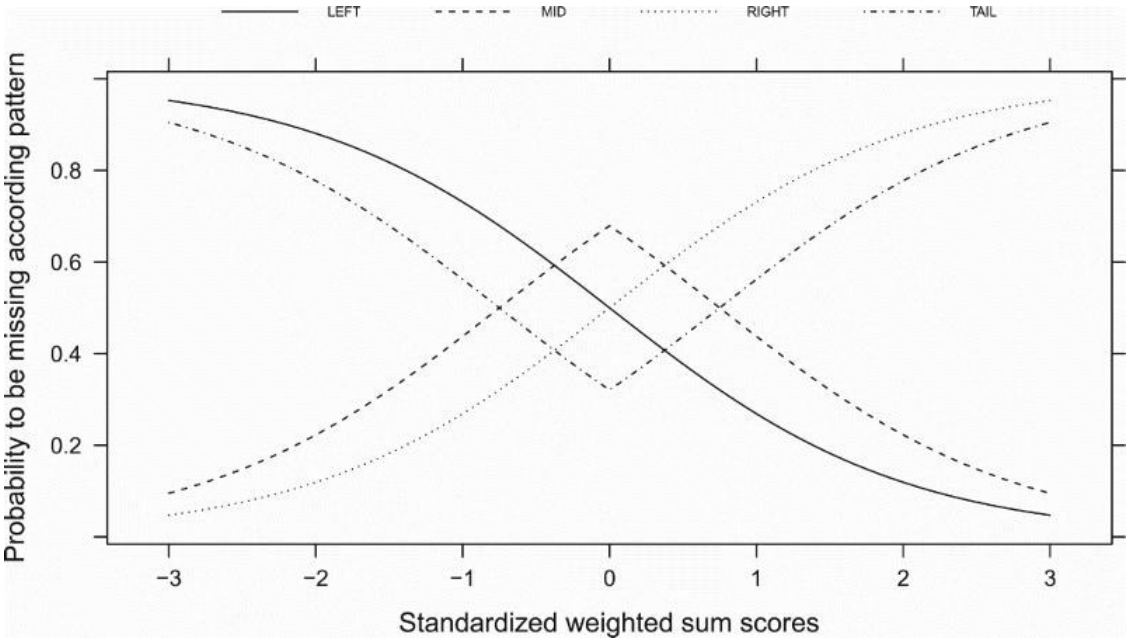

Figura 3.7: Diferentes distribuições possíveis para a determinação de probabilidades para cada peso. [25]

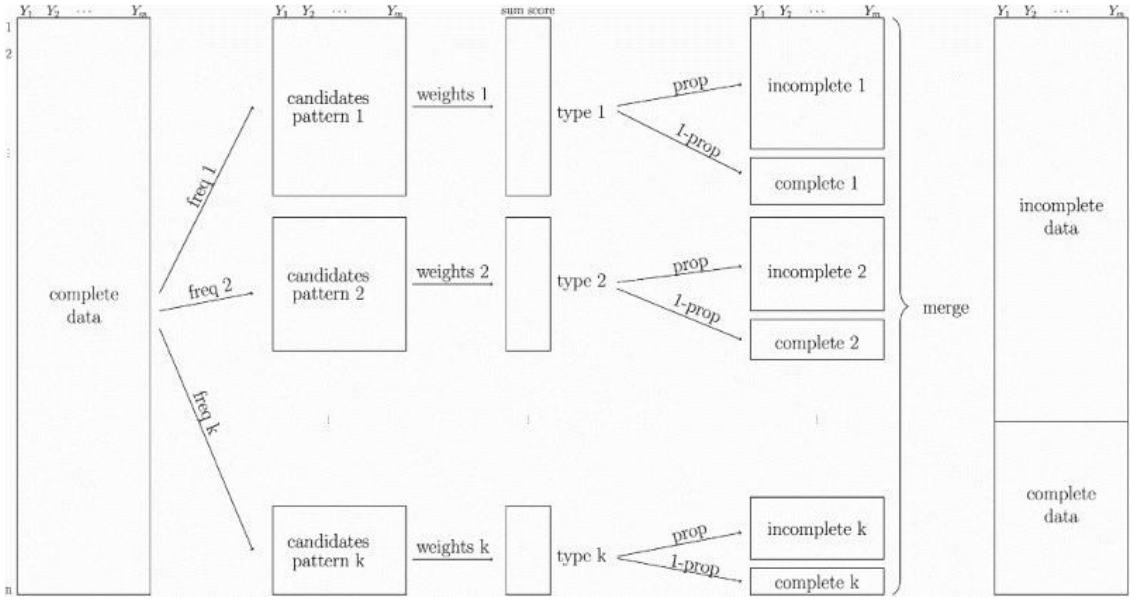

Figura 3.8: Esquema representativo do processo de criação de valores em falta [25]

3.4.3 Avaliação

Quanto à avaliação feita aos algoritmos de imputação de valores em falta. Utilizaram-se o dataset original e os datasets com valores imputados através da previsão feita por cada algo algoritmo. Para cada algoritmo é calculado o MSE, usando o dataset original e o seu dataset com valores imputados, fazendo o quadrado da diferença entre os valores reais e os valores previstos, dividindo-se no fim pelo número de valores do dataset. Após o cálculo do MSE (equação 7) é feita a raiz quadrada deste valor obtendo assim o RMSE (equação 8). Os algoritmos são ordenados por ordem crescente do RMSE, sendo o primeiro o melhor. Esta medida de avaliação foi escolhida pois é uma das medidas mais comuns (encontrada em diferente artigos) usada para este tipo de problema.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \qquad (7)$$

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}} \qquad (8)$$

Capítulo 4

Aplicações

4.1 Introdução

Neste capítulo é explicado as aplicações desenvolvidas. Está incluído a especificação da interface de pré-processamento de dados desenvolvida para o projeto POWER, isto é, são mostradas as funções que constituem este módulo, sendo explicado o funcionamento de cada uma. Juntamente com isto são explicados os parâmetros de entrada e a variável que é retornada. Também são mostrados alguns exemplos do seu funcionamento.

Para além da interface a ser entregue, é também mostrada a aplicação desenvolvida para mostrar o funcionamento do módulo criado. É feita uma explicação dos casos de uso da aplicação e da sua arquitetura. No fim são mostradas todas as interações possíveis com a aplicação.

4.2 Interface de pré-processamento de dados

4.2.1 Especificação da interface

Juntamente com o módulo de pré-processamento de dados , houve a necessidade de desenvolver um documento com a especificação do módulo. Isto é, um documento onde são explicadas a funções que contém, para assim facilitar a sua consulta e utilização por parte de outros membros do projeto. Algumas dessas funções são mostradas aqui, sendo as restantes apresentadas no apêndice A.

pipeline_bp(data, patient_id)

Retorna um array de dimensão igual ao que é dado como entrada. Quando os parâmetros são inválidos é retornada uma mensagem de erro. Caso os parâmetros sejam válidos, a função realiza a deteção de anomalias e sua eliminação, seguida da imputação de valores em falta, retornando um array com os dados corrigidos. Usada para a situação de pressão sanguínea,

Parâmetros

- **data : array_like**

  Array com dados de input. Um valor de pressão sanguínea por dia, acompanhada de dados adicionais (por exemplo o ritmo cardíaco da pessoa). Valores devem ser int ou float.

- **patiend_id : None ou int**

  O parâmetro do id do paciente pode estar vazio (None) ou pode ser um número inteiro. Usado para aceder rankings específicos do paciente, caso existam.

Output

- **result : array_like**

  Array com tamanho igual ao de array de entrada, contendo os dados sem anomalias e valores em falta.

Notas

Caso seja introduzido o id do paciente é feita a procura do ranking do paciente através da função get_rankings. Caso não seja especificado um id, são obtidos os rankings respetivos à situação da pressão sanguínea, que foram previamente definidos.

pipeline_glicose_training(data, patient_id)

Retorna um dicionário com os rankings. Quando os parâmetros são inválidos é retornada uma mensagem de erro. Caso os parâmetros sejam válidos, a função realiza a simulação da deteção de anomalias e a simulação da imputação de valores em falta, retornando uma dicionário com os rankings e guarda um ficheiro com os rankings. Usada para a situação de glicose.

Parâmetros

- **data : array_like**

  Array com dados de input. Um valor de glicose por cada 5 minutos, acompanhada de dados adicionais (por exemplo o peso da pessoa). Valores devem ser int ou float.

- **patiend_id : None ou int**

  O parâmetro do id do paciente pode estar vazio (None) ou pode ser um número inteiro. Usado para aceder rankings específicos do paciente, caso existam.

Output

- **result : dict**

  Dicionário com os rankings. Dividido por deteção de anomalias e imputação de valores em falta, cada um com os seus algoritmos e cada algoritmo com os seu rankings.

Notas

Caso seja introduzido o id do paciente é guardado um ranking para esse paciente. Caso não seja especificado um id, é guardado o ranking para as variáveis utilizadas, usando a função save_rankings.

4.2.2 Testes

Para a realização desta interface foi criado um conjunto de testes para assegurar o seu bom funcionamento. A seguinte lista contém alguns dos testes realizados para a interface desenvolvida. Os restantes testes podem ser consultados no apêndice B.

Id do teste: 1

Descrição do teste: Verificar se a pipeline de ranking retorna o tipo correto de dados (array de dados)

Passos do teste:

- Passar os dados do dataset completo pela pipeline
- Verificar se o tipo e dado retornados é o pretendido

Dados usados para teste:

- Dataset completo

Resultado expectável: Variável do tipo array

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

Id do teste: 2

Descrição do teste: Verificar se os rankings ficam guardados da maneira correta

Passos do teste:

- Passar uma lista de rankings pelo algoritmo
- Verificar se um novo ficheiro foi guardado
- Inspecionar o conteúdo do ficheiro

Dados usados para teste:

- Lista de Rankings

Resultado expectável: Conteúdo do ficheiro respeita o formato definido

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

Id do teste: 3

Descrição do teste: Verificar se após a previsão dos valores em falta, esses valores ficam dentro de limites conhecidos

Passos do teste:

- Passar o dataset incompleto por cada algoritmo
- Verificar o resultado de cada algoritmo

Dados usados para teste:

- Dataset incompleto

Resultado expectável: Valores imputados não excedem os valores máximos e mínimos conhecidos para pressão sanguínea e glicose

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

Id do teste: 4

Descrição do teste: Verificar se a percentagem de dados em falta respeita a percentagem que foi selecionada previamente

Passos do teste:

- Criar um dataset incompleto
- Verificar se a percentagem de valores em falta é igual ou próxima da introduzida

Dados usados para teste:

- Dataset completo
- Percentagens de valores em falta (10, 20, 30)

Resultado expectável: A percentagem de valores em falta é igual à passada como input

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

4.3 Aplicação demonstrativa

4.3.1 Requisitos

Nesta aplicação é pretendido a criação de uma interface que permita a interação com o módulo pré-processamento de dados, de forma se puder visualizar o seu funcionamento de uma maneira mais interativa e interessante, sendo o seu objetivo final, complementar a demonstração do funcionamento do módulo durante a

apresentação desta dissertação. Para isso houve a necessidade de desenvolver uma arquitetura e extrair requisitos através de um cenário concreto.

4.3.1.1 Descrição da aplicação

A aplicação realizada pretende servir de frontend para o módulo de pré-processamento de dados criado para o projeto POWER. Com esta aplicação pretende-se mostrar o funcionamento de todas a componentes que foram criadas, isto é, mostrar o processo de deteção de outliers e sua eliminação e posteriormente a imputação de valores em falta já existentes e que surjam da eliminação dos outliers.

Pretende-se permitir ao utilizador que selecione os dados que pretende usar para teste. Permitindo-lhe selecionar as variáveis que se pretendem usar. Após esta seleção, são mostrados os dados num gráfico e também são mostrados todos os algoritmos para deteção de outliers e de imputação de valores em falta, ordenados por rank para a situação específica, podendo ser um ranking geral para a combinação de variáveis (previamente estabelecido à criação da aplicação) ou caso seja um paciente, cujo o seu id já se encontra registado e tenha sido feito um ranking específico para esse paciente. A seguir à seleção dos algoritmos (caso não sejam selecionados, são utilizados os que estão em primeiro lugar), o utilizador pode selecionar para que a aplicação efetuar a deteção de outliers e subsequente eliminação e depois a imputação dos valores em falta, mostrando depois no gráfico todas as alterações efetuadas.

Por fim, a aplicação permite também ao utilizador que faça a criação de um ranking específico para um paciente. Para este caso é necessário que os dados estejam completos. No fim deste processo é criado um novo ranking que pode ser usado posteriormente para efetuar o pré-processamento dos dados desse paciente.

4.3.1.2 Casos de Uso

Na figura 4.1 é possível visualizar um diagrama com os casos de uso da aplicação demonstrativa. Estes casos de uso representam as funções descritas anteriormente. Uma breve descrição de cada caso de uso é apresentado na tabela 4.1.

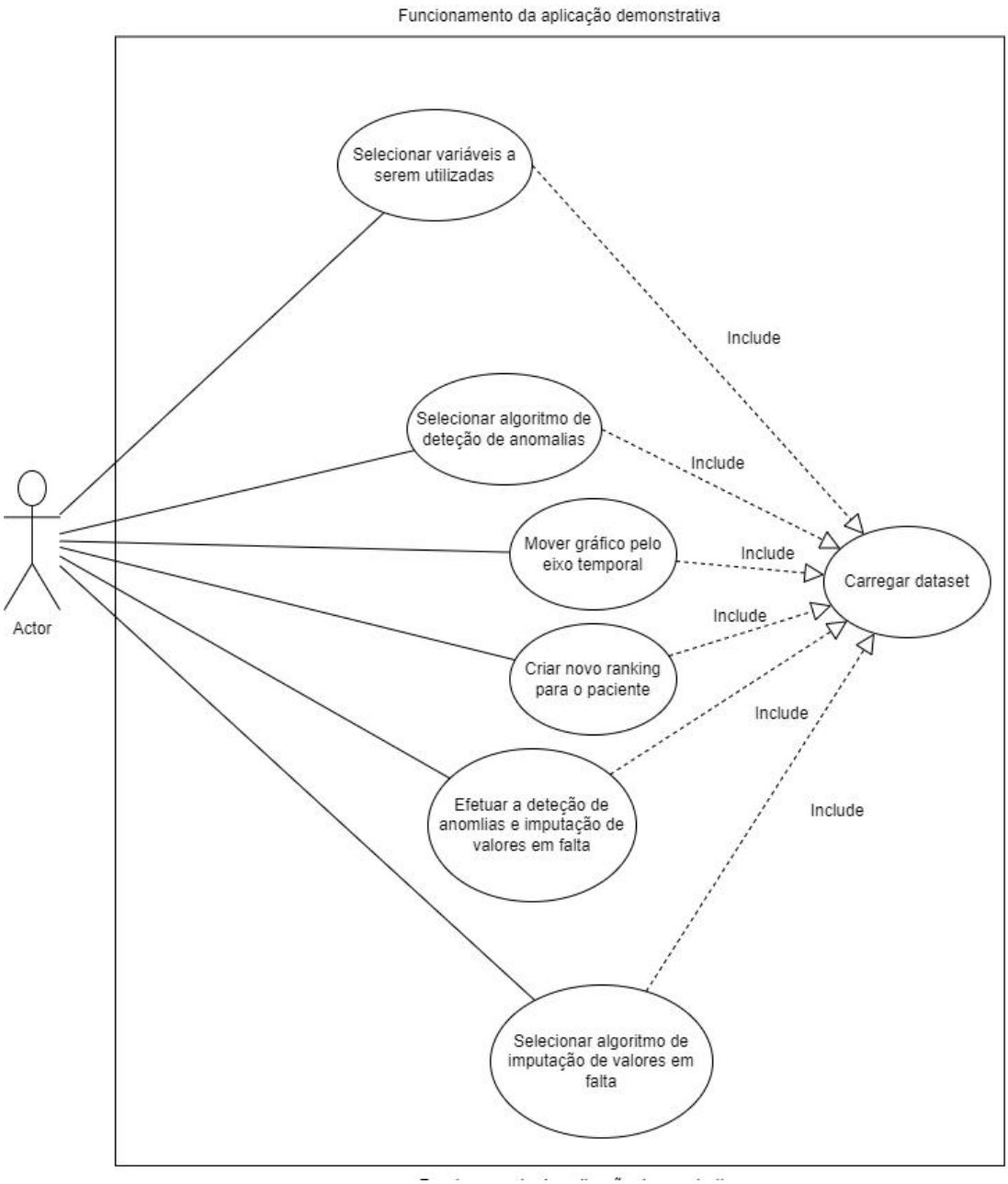

Figura 4.1: Diagrama de casos de uso da aplicação demonstrativa

Caso de uso	Descrição do caso de uso
Carregar Dataset	O utilizador deve ser capaz de selecionar um dataset existente e carregá-lo para a aplicação. Os dados devem ser mostrados no gráfico.
Selecionar variáveis a serem utilizadas	O utilizador pode selecionar que variáveis serão consideradas, sendo as outras eliminadas (ignoradas)

	pelo sistema. Caso nenhuma seja selecionada, considera-se o dataset inteiro.
Selecionar algoritmo para deteção de anomalias	O utilizador pode selecionar o algoritmo de deteção de anomalias a ser utilizador. Caso não seja utilizado, é usado o que tem o primeiro rank.
Selecionar algoritmo para imputação de valores em falta	O utilizador pode selecionar o algoritmo de deteção de anomalias a ser utilizador. Caso não seja utilizado, é usado o que tem o primeiro rank.
Efetuar deteção de anomalias e imputação de valores em falta	Após todas as seleções terem sido feitas, o utilizador pode pedir ao sistema que efetue a deteção de anomalias e a sua eliminação e a imputação dos valores em falta. Os resultados devem ser mostrados no gráfico.
Criar novo ranking para o paciente	O utilizador tem de ser capaz de criar um novo ranking para o paciente a quem pertencem os dados. O ranking deve depois ser guardado.
Mover gráfico pelo eixo temporal	O utilizador deve ser capaz de visualizar dados que estejam em tempos posteriores ou passados aos que são mostrados no gráfico.

Tabela 4.1: Descrição dos casos de uso da aplicação demonstrativa

4.3.2 Arquitetura

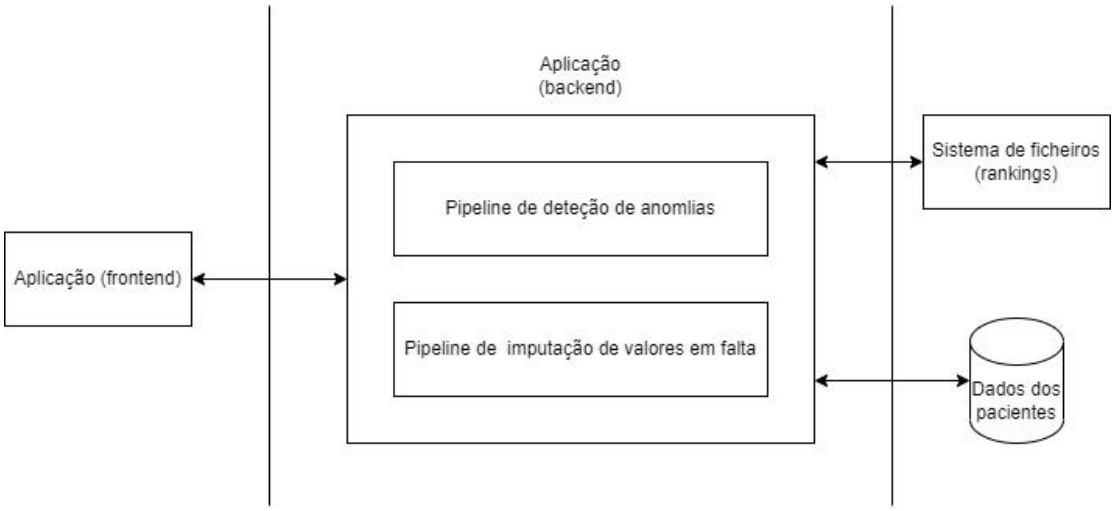

Figura 4.2: Arquitetura da aplicação demonstrativa

A arquitetura proposta para o sistema está dividida em três partes, representada pela figura 4.2. Começando pela esquerda, a primeira secção corresponde ao frontend da aplicação, onde está criadas as interfaces que permitem que utilizador possa realizar as ações descritas nos casos de uso (figura 4.1).

O frontend comunica com o backend, onde estão inseridas as pipelines responsáveis por fazer o pré-processamento dos dados, isto é, a pipeline deteção de anomalias e sua eliminação e a pipeline de imputação de valores em falta. O backend também tem a capacidade de fazer a criação de rankings e o seu acesso.

Por fim, o backend comunica com a base de dados onde são guardados os dados dos pacientes e que são utilizados pelo utilizador para testar o funcionamento do módulo. Também é feito o acesso a ficheiros onde estão guardados os rankings dos algoritmos e onde são guardados outros rankings que sejam criados pelo utilizador.

4.3.3 Interações da aplicação

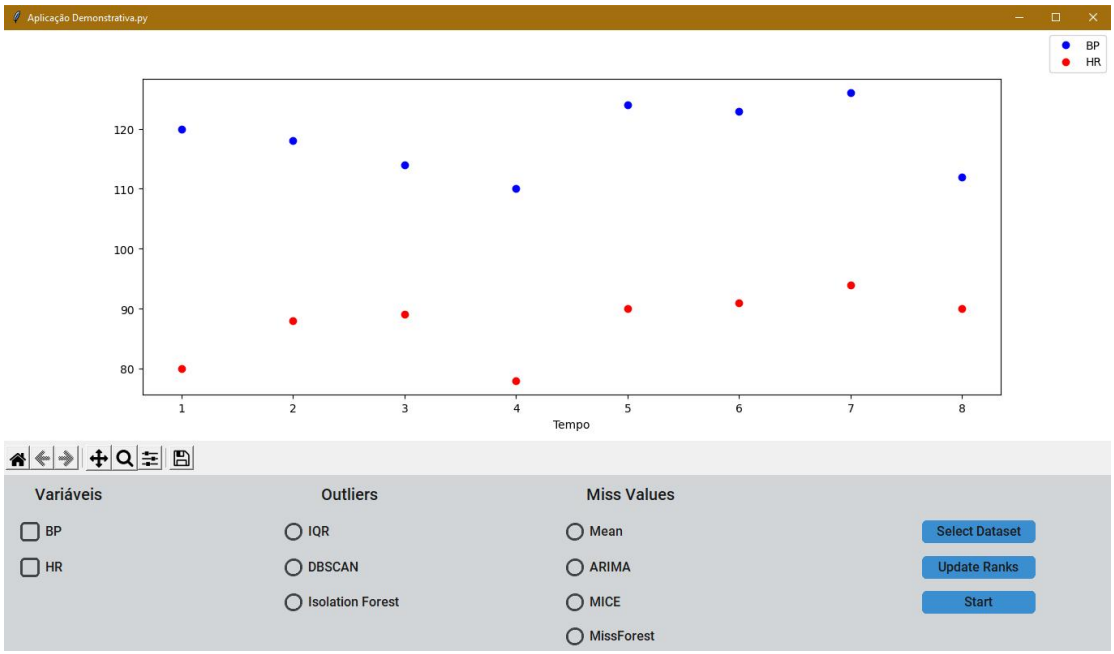

Figura 4.3: Aplicação Demonstrativa

A figura 4.3 representa o frontend da aplicação demonstrativa criada para mostrar o funcionamento do módulo criado neste projeto.

O utilizador começa por clicar no botão "Select Dataset", lhe mostra uma nova janela com os possíveis datasets que se encontram guardados. Após a seleção do dataset, as variáveis que se encontram presentes no dataset são mostradas na coluna de variáveis e cada uma das colunas com os algoritmos ficam com os seus algoritmos ordenados tendo em conta as variáveis que estejam presentes, ou caso o paciente já tenha um ranking, permite-lhe que tenha os algoritmos ordenados para o seu caso em específico.

Depois podemos ter duas situações diferentes. Caso o dataset seja incompleto o utilizador pode selecionar as variáveis que pretende que sejam utilizadas e pode selecionar o botão de update ranks para realizar a simulação das situações de deteção de anomalias e de imputação de valores em falta, criando no processo um novo dataset só com essas variáveis e um novo ranking para essa situação e paciente guardando o dataset e ranking obtidos, para serem utilizados no futuro. Caso o dataset seja incompleto pode-se selecionar também as variáveis que queremos que sejam usadas e os algoritmos que pretendemos para cada situação. Caso não sejam

selecionados algoritmos, consideram-se os que estão em primeiro lugar no ranking. Após a seleção do botão start é feita a deteção de anomalias e a sua eliminação e a imputação dos valores em falta usando os algoritmos escolhidos. O gráfico acima das opções é alterado para representar os resultados obtidos.

No gráfico é possível efetuar o efetuar zoom em certas zonas do gráfico com o botão da lupa. E com o botão das quatro setas é possível mover o gráfico. Com os dois botões de setas é possível andar para frente ou para trás nas alterações que foram feitas ao gráfico com os dois botões anteriores. O botão com a casa coloca o gráfico na sua forma original.

4.4 Conclusão

Desta secção pode-se retirar o seguinte. Foram mostrados as principais componentes do módulo que foi criado, isto é, as funções que constituem o seu funcionamento. Depois são mostrados alguns dos testes que foram desenvolvidos para garantir o funcionamento correto do módulo. Na segunda secção deste capítulo é feita uma explicação dos requisitos e casos de uso referentes à aplicação demonstrativa, que pretende mostrar o funcionamento do módulo de uma forma mais interativa e interessante. Em seguida é mostrada a arquitetura desta aplicação, que incluí o frontend, o backend e os locais onde são guardados os dados dos pacientes e dos rankings. Por fim é mostrada aplicação e as interações possíveis de se realizarem.

Capítulo 5

Resultados e Discussão

5.1 Introdução

Este capítulo visa mostrar os resultados de cada uma das interfaces desenvolvidas também são expostas explicações que pretendem oferecer possíveis deduções para o porquê de cada um dos resultados obtidos.Tendo por base o conhecimento obtido pela literatura consultada para o desenvolvimento desta dissertação e os resultados obtidos.

5.2 Dados em falta

5.2.1 Resultados

As seguintes tabelas mostram os resultados obtidos para o caso da imputação de valores em falta. Utilizaram-se diferentes combinações de variáveis e quantidades de valores em falta. Para os casos univariáveis, utilizaram-se somente os algoritmos que são específicos a esses casos, dado que os algoritmos para situações multivariáveis não funcionam nestas situações Os restantes resultados podem ser consultados no apêndice C.

	Percentagem de Valores em falta		
Algoritmo	**10**	**20**	**30**
Imputação com Média	0.470	**1.067**	1.642
ARIMA	**0.414**	1.230	**1.577**

Tabela 5.1: Resultados da imputação para dados com BP

	Percentagem de Valores em falta		
Algoritmo	**10**	**20**	**30**
Imputação com Média	0.202	0.551	0.596
ARIMA	0.201	0.544	0.579
Knn	0.226	0.500	0.573
MICE	0.280	0.514	0.768
MissForest	**0.121**	**0.400**	**0.489**
VAR-IM	0.315	0.520	0.604

Tabela 5.2: Resultados da imputação para dados com BP, BW, BR e HR

	Percentagem de Valores em falta		
Algoritmo	**10**	**20**	**30**
Imputação com Média	0.557	0.647	0.750
ARIMA	0.565	0.641	0.739
Knn	**0.470**	0.665	0.749
MICE	0.630	0.708	**0.707**
MissForest	0.598	**0.628**	0.753
VAR-IM	0.738	0.791	0.893

Tabela 5.3: Resultados da imputação para dados com BP, BR e HR

	Percentagem de Valores em falta		
Algoritmo	**10**	**20**	**30**
Imputação com Média	0.356	1.547	1.825
ARIMA	0.357	1.587	1.851
Knn	**0.341**	1.595	1.802
MICE	0.471	1.521	1.847
MissForest	0.525	**1.505**	**1.785**
VAR-IM	0.441	1.752	2.150

Tabela 5.4: Resultados da imputação para dados com BP e HR

	Percentagem de Valores em falta		
Algoritmo	**10**	**20**	**30**
Imputação com Média	4.407	6.396	7.956
ARIMA	**0.338**	**0.525**	**0.826**

Tabela 5.5: Resultados da imputação para dados com glicose

A parir dos resultados obtidos, podes-se verificar que para as situações multivariáveis, que os algoritmos para situações multivariáveis foram os que obtiveram, na maioria,

os melhores resultados (assinalados a verde). Os algoritmos univariáveis conseguiram resultados próximos dos resultados dos algoritmos multivariáveis, sendo que obtiveram melhores resultados em situações em que haviam menos valores em falta e menos variáveis.

Nas situações univariáveis o melhor algoritmo foi o ARIMA, com maior discrepância no caso da glicose.

Algo que também se pode observar no caso multivariável, é o facto de para diferentes variáveis e diferentes percentagens de valores em falta pode-se observar que se obteve diferentes algoritmos considerados como o melhor da sua situação. Sendo que o algoritmo MissForest obteve a maioria desses valores.

O algoritmo VAR-IM acabou por obter a maioria dos piores resultados.

5.2.2 Discussão

Relativamente ao facto dos algoritmos multivariáveis terem obtido melhores resultados nas situações em que o dataset contém mais que uma variável. Isto pode ser explicado pelo simple facto dos algoritmos multivariáveis procurarem fazer a imputação de valores em falta ao utilizarem outras variáveis. Ou seja, têm em consideração possíveis relações que possam existir entre a variáveis, criando resultados menos enviesados e que mantêm as relações entre as variáveis. Algo que não é possível com algoritmos univariáveis.

Quanto ao facto de se obterem diferente algoritmos como melhores para diferentes situações. Isto pode-se dever à seleção dos parâmetros ter levado a que alguns algoritmos se comportassem de uma maneira melhor para certas situações e noutros casos obtivessem resultados piores. A seleção dos parâmetros é um fator bastante importante para determinar o bom ou mau funcionamento de um algoritmo. E cada situação pode requerer que os parâmetros de um determinado algoritmo sejam alterados. O facto de ser possível obterem-se diferentes algoritmos como melhores, suporta a hipótese da criação de um sistema de rankings para situações específicas.

O mau funcionamento do algoritmo VAR-IM pode-se ter devido a uma seleção incorreta do parâmetro de atraso. Contudo a explicação mais plausível é que a

implementação do algoritmo pode ter sido feita de maneiro incorreta, derivada de uma má interpretação do artigo.

Quanto á situação univariável, era já esperado que o algoritmo ARIMA obtive-se melhores resultados , dado que é um algoritmo específico para ser usado em séries temporais univariáveis. No entanto pode-se observar que a diferença no caso do dataset com os dados de BP, o IQR não ficou muito afastado. Isto pode-se dever ao facto de o dataset ser pequeno (45 valores) e dos valores de BP não apresentarem muita variação. Isto já não se verifica no caso do dataset da glicose, em que os resultados ficaram muito diferentes.

Em termos de percentagem de valores em falta, os algoritmos univariáveis conseguiram resultados mais próximos dos algoritmos multivariáveis quando a percentagem de valores em falta é menor e há menos variáveis. Isto pode-se dever ao facto de que em situações com menos valores em falta e menos variáveis, as consequências negativas destes algoritmos não sejam tão predominantes. O que sugere que em situações mais simples este algoritmos possam até ser utilizados, evitando-se utilizar algoritmos mais complexos.

5.3 Outlier

5.3.1 Resultados

As seguintes tabelas mostram os resultados obtidos para o caso da deteção de anomalias. Utilizaram-se diferente combinações de variáveis, de percentagem de anomalias e percentagem de incremento para tornar o valor anómalo. Os restantes resultados podem ser consultados no apêndice C.

	Percentagem de Outliers\Incremento do outlier (%)								
Algoritmo	**5\10**	**5\20**	**5\30**	**10\10**	**10\20**	**10\30**	**15\10**	**15\20**	**15\30**
IQR	**0.966**	**0.964**	**0.976**	**0.963**	**0.976**	0.895	0.500	**0.975**	**0.963**
DBSCAN	0.500	0.500	0.500	0.500	0.500	**1.000**	0.550	0.587	0.500
Isolation Forest	0.830	0.821	0.881	0.841	0.821	0.860	**0.679**	0.875	0.841

Tabela 5.6: Resultados da deteção de anomalias para dados com BP

	Percentagem de Outliers\Incremento do outlier (%)								
Algoritmo	**5\10**	**5\20**	**5\30**	**10\10**	**10\20**	**10\30**	**15\10**	**15\20**	**15\30**
IQR	0.714	0.881	0.881	0.753	0.878	0.878	0.500	**0.900**	**0.900**
DBSCAN	0.631	0.786	0.786	0.588	0.826	0.826	**0.693**	0.793	0.793
Isolation Forest	**0.893**	**0.893**	**0.893**	**0.902**	**0.902**	**0.915**	0.664	0.743	0.821

Tabela 5.7: Resultados da deteção de anomalias para dados com BP, HR, BW e BR

	Percentagem de Outliers\Incremento do outlier (%)								
Algoritmo	**5\10**	**5\20**	**5\30**	**10\10**	**10\20**	**10\30**	**15\10**	**15\20**	**15\30**
IQR	0.895	0.895	0.895	0.777	**0.902**	0.902	0.664	0.789	0.851
DBSCAN	0.727	**0.977**	**0.977**	0.726	0.851	0.851	**0.674**	**0.799**	0.799
Isolation Forest	**0.895**	0.907	0.907	**0.802**	0.826	**0.963**	0.620	0.758	**0.924**

Tabela 5.8: Resultados da deteção de anomalias para dados com BP, BR e HR

	Percentagem de Outliers\Incremento do outlier (%)								
Algoritmo	**5\10**	**5\20**	**5\30**	**10\10**	**10\20**	**10\30**	**15\10**	**15\20**	**15\30**
IQR	**0.863**	0.963	0.963	**0.863**	0.963	0.963	0.708	0.962	0.986
DBSCAN	0.800	**1.000**	**1.000**	0.800	**1.000**	**1.000**	0.611	**0.984**	**1.000**
Isolation Forest	0.775	0.925	0.938	0.775	0.925	0.938	**0.708**	0.915	0.944

Tabela 5.9: Resultados da deteção de anomalias para dados com BP e HR

	Percentagem de Outliers\Incremento do outlier (%)								
Algoritmo	**5\10**	**5\20**	**5\30**	**10\10**	**10\20**	**10\30**	**15\10**	**15\20**	**15\30**
IQR	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
DBSCAN	0.520	0.605	0.605	0.553	0.631	0.653	0.563	0.633	0.671
Isolation Forest	**0.703**	**0.734**	**0.734**	**0.754**	**0.755**	**0.827**	**0.705**	**0.770**	**0.805**

Tabela 5.10: Resultados da deteção de anomalias para dados com glicose

A partir da tabela de dados de BP, observou-se que o algoritmo IQR obteve os melhores resultados, na maioria das situações. No entanto na situação da glicose os resultados foram muito piores.

Em relação ao caso multivariável observou-se que os algoritmos multivariáveis obtiveram a maioria dos resultados melhores, sendo que o IQR manteve-se próximo.

Também pode-se observar que os resultados, exceto na situação da glicose, variam. Isto o melhor algoritmo altera-se algumas vezes dependendo da situação.

Quanto à percentagem de anomalias não se notam diferenças de performance, mas nota-se quando se altera a percentagem de incremento do valor anómalo.

5.3.2 Discussão

O facto do IQR obter melhores resultados na situação univariável, era expectável, dado que é um algoritmo univariável. Sendo que os outros funcionam melhor quando existem outras variáveis presentes No entanto no caso multivariável a situação iinverte-se, mostrando que os algoritmos multivariáveis funcionam melhor na sua situação específica, dado que têm em consideração todas as variáveis na deteção de anomalias.

O facto dos melhores algoritmos variarem, pode ser causado pela seleção dos parâmetros de cada algoritmo. Isto é, se diferentes parâmetros fossem escolhidos seria expectável obter diferentes resultados e possivelmente até diferentes algoritmos como melhores. Os valores escolhidos como anomalias e as variáveis utilizadas, podem também influenciar estes resultados. Isto sustenta a ideia da criação de rankings diferentes para situações específicas.

O algoritmos têm pior performance quando o valor de incremento é menor. Isto era esperado, visto que se o valor fica pouco afastado dos restantes valores, este acaba por ser mais difícil de ser detetado e acaba por se disfarçar melhor, escapando assim à deteção por parte dos algoritmos.

Quanto ao caso da glicose, os resultados foram piores, devido à presença de uma maior quantidade de dados e de uma maior variância de valores, traduzindo-se no caso que se obteve.

5.4 Conclusão

As principais situações que se podem retirar deste capítulo, é o facto de que os algoritmos multivariáveis obtiveram melhores resultados na maioria dos casos em que o os datasets tinham mais que uma variável. Sendo que esta discrepância foi mais notória quando o número de variáveis e de valores em falta foi maior. No caso univariável observou-se o oposto, sendo que só foi possível na deteção de anomalias.

Observou-se também que existe alguma variação quanto ao algoritmo que é considerado melhor, fomentando a hipótese da criação de um sistema de rankings para casos específicos, que podem ser selecionados automaticamente.

Capítulo 6

Conclusão e trabalho futuro

Neste trabalho foi desenvolvida uma investigação e seleção de possíveis algoritmos para resolverem os problemas de deteção de anomalias e de imputação de valores em falta em séries temporais com valores de pressão sanguínea e de glicose. Explorando situações mutivariáveis e univariáveis para ambos os problemas. Procurou-se determinar os melhores algoritmos para cada situação, tendo em conta as variáveis presentes e as quantidades de valores em falta e de anomalias, verificando se este algoritmos selecionados variavam. Neste processo procurou-se também determinar se os algoritmos multivariáveis apresentavam melhores resultados para as situações em que os dados continham mais que uma variável, em comparação com algoritmos univariáveis.

Chegou-se à conclusão que os algoritmos multivariáveis obtiveram melhores resultados para as situações em que o dataset era multivariável. Observou-se também que o algoritmo melhor pode variar dependendo das variáveis utilizadas e da quantidade de valores em falta e de anomalias que existam nos dados. Posto isto, foi então desenvolvido um módulo que permite realizar a deteção de anomalias e imputação de valores em falta usando algoritmos que se encontram organizados em rankings que são selecionados automaticamente tendo em conta a situação que se pretende resolver. O módulo também permite a criação de novos rankings ao se usar datasets completos em simulações destes problemas.

Tendo em conta o trabalho desenvolvido, existem pontos a serem melhorados. Pretende-se testar este módulo com um dataset maior e contenha as variáveis a ser utilizadas no projeto POWER. Também terão de ser feitas alterações ao módulo de modo a que possa ser integrado no projeto com os restantes módulos que estão a ser criados por outros colegas.

Referências

[1] Moritz S, Bartz-Beielstein T. imputeTS: time series missing value imputation in R. R J.. 2017 Jun 1;9(1):207.

[2] KANG, Hyun. The prevention and handling of the missing data. Korean journal of anesthesiology, 2013, 64.5: 402-406.

[3] Maddala, G.S. (2013), "Introduction to Econometrics" (2nd edition)

[4] Divya D. and S. S. Babu, "Methods to detect different types of outliers," 2016 International Conference on Data Mining and Advanced Computing (SAPIENCE), 2016, pp. 23-28, doi: 10.1109/SAPIENCE.2016.7684114.

[5] Cousineau, Denis & Chartier, Sylvain. (2010). Outliers detection and treatment: A review. International Journal of Psychological Research. 3. 10.21500/20112084.844.

[6] Rousseeuw, P.J. and Hubert, M., 2011. Robust statistics for outlier detection. Wiley interdisciplinary reviews: Data mining and knowledge discovery, 1(1), pp.73-79.

[7] Zimek, A., & Filzmoser, P. (2018). There and back again: Outlier detection between statistical reasoning and data mining algorithms. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(6), [e1280],https://doi.org/10.1002/widm.1280.

[8] Donders AR, Van Der Heijden GJ, Stijnen T, Moons KG. A gentle introduction to imputation of missing values. Journal of clinical epidemiology. 2006 Oct 1;59(10):1087-91.

[9] Schafer, J. L. (1999) 'Multiple imputation: a primer', Statistical Methods in Medical Research, 8(1), pp. 3–15. doi: 10.1177/096228029900800102.

[10] Grigorios Papageorgiou, Stuart W Grant, Johanna J M Takkenberg, Mostafa M Mokhles, Statistical primer: how to deal with missing data in scientific research?, Interactive CardioVascular and Thoracic Surgery, Volume 27, Issue 2, August 2018, Pages 153–158, https://doi.org/10.1093/icvts/ivy102

[11] "Power - Empowering a digital future"; [Online]; Available: https://www.cisuc.uc.pt/en/projects/power

[12] LIANG, Jiongqian; PARTHASARATHY, Srinivasan. Robust contextual outlier detection: Where context meets sparsity. In: Proceedings of the 25th ACM international on conference on information and knowledge management. 2016. p. 2167-2172.

[13] Wijaya, C.Y. "Outlier — Why is it important?"; [Online], consultado a 4 de Agosto de 2022, https://towardsdatascience.com/outlier-why-is-it-important-af58adbefecc

[14] Kaiser, R. and Maravall Herrero, A., 1999. Seasonal outliers in time series. Documentos de trabajo/Banco de España, 9915.

[15] ESTER, Martin, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd. 1996. p. 226-231.

[16] GAN, Guojun; NG, Michael Kwok-Po. K-means clustering with outlier removal. Pattern Recognition Letters, 2017, 90: 8-14.

[17] https://stefvanbuuren.name/fimd/

[18] Azur, M. J., Stuart, E. A., Frangakis, C., & Leaf, P. J. (2011). Multiple imputation by chained equations: what is it and how does it work?. International journal of methods in psychiatric research, 20(1), 40–49. https://doi.org/10.1002/mpr.329

[19] T. Mahboob, A. Ijaz, A. Shahzad and M. Kalsoom, "Handling Missing Values in Chronic Kidney Disease Datasets Using KNN, K-Means and K-Medoids Algorithms," 2018 12th International Conference on Open Source Systems and Technologies (ICOSST), 2018, pp. 76-81, doi: 10.1109/ICOSST.2018.8632179.

[20] BASHIR, Faraj; WEI, Hua-Liang. Handling missing data in multivariate time series using a vector autoregressive model-imputation (VAR-IM) algorithm. Neurocomputing, 2018, 276: 23-30.

[21] Daniel J. Stekhoven, Peter Bühlmann, MissForest—non-parametric missing value imputation for mixed-type data, Bioinformatics, Volume 28, Issue 1, 1 January 2012, Pages 112–118, https://doi.org/10.1093/bioinformatics/btr597

[22] LI, Yuebiao; LI, Zhiheng; LI, Li. Missing traffic data: comparison of imputation methods. IET Intelligent Transport Systems, 2014, 8.1: 51-57.

[23] ÇELIK, Mete; DADAŞER-ÇELIK, Filiz; DOKUZ, Ahmet Şakir. Anomaly detection in temperature data using DBSCAN algorithm. In: 2011 international symposium on innovations in intelligent systems and applications. IEEE, 2011. p. 91-95.

[24] WALFISH, Steven. A review of statistical outlier methods. Pharmaceutical technology, 2006, 30.11: 82.

[25] Rianne Margaretha Schouten, Peter Lugtig & Gerko Vink (2018) Generating missing values for simulation purposes: a multivariate amputation procedure, Journal of Statistical Computation and Simulation, 88:15, 2909-2930, DOI: 10.1080/00949655.2018.1491577

[26] LIU, Fei Tony; TING, Kai Ming; ZHOU, Zhi-Hua. Isolation forest. In: 2008 eighth ieee international conference on data mining. IEEE, 2008. p. 413-422.

[27] BIAU, Gérard; SCORNET, Erwan. A random forest guided tour. Test, 2016, 25.2: 197-227.

[28] JEBB, Andrew T., et al. Time series analysis for psychological research: examining and forecasting change. Frontiers in psychology, 2015, 6: 727.

[29] "O Povo Algarvio"; [Online]; consultado a 30 de Agosto de 2022 https://algarveproperty.pt/algarve/o-povo-algarvio-residentes-no-algarve-populacao-do-algarve-populacao-algarvia_77

[30] LAZAREVIC, Aleksandar; KUMAR, Vipin. Feature bagging for outlier detection. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. 2005. p. 157-166.

Apêndice A - Especificação da Interface

pipeline_glicose(data, patient_id)

Retorna um array de dimensão igual ao que é dado como entrada. Quando os parâmetros são inválidos é retornada uma mensagem de erro. Caso os parâmetros sejam válidos, a função realiza a deteção de anomalias e sua eliminação, seguida da imputação de valores em falta, retornando um array com os dados corrigidos. Usada para a situação de glicose.

Parâmetros

- **data : array_like**

  Array com dados de input. Um valor de glicose por cada 5 minutos, acompanhada de dados adicionais (por exemplo o peso da pessoa). Valores devem ser int ou float.

- **patiend_id : None ou int**

  O parâmetro do id do paciente pode estar vazio (None) ou pode ser um número inteiro. Usado para aceder rankings específicos do paciente, caso existam.

Output

- **result : array_like**

  Array com tamanho igual ao de array de entrada, contendo os dados sem anomalias e valores em falta.

Notas

Caso seja introduzido o id do paciente é feita a procura do ranking do paciente através da função get_rankings. Caso não seja especificado um id, são obtidos os rankings respetivos à situação da glicose, que foram previamente definidos.

pipeline_bp_training(data, patient_id)

Retorna um dicionário com os rankings. Quando os parâmetros são inválidos é retornada uma mensagem de erro. Caso os parâmetros sejam válidos, a função realiza a simulação da deteção de anomalias e a simulação da imputação de valores em falta, retornando uma dicionário com os rankings e guarda um ficheiro com os rankings. Usada para a situação de pressão sanguínea.

Parâmetros

- **data : array_like**

  Array com dados de input. Um valor de pressão sanguínea por dia, acompanhada de dados adicionais (por exemplo o ritmo cardíaco da pessoa). Valores devem ser int ou float.

- **patiend_id : None ou int**

  O parâmetro do id do paciente pode estar vazio (None) ou pode ser um número inteiro. Usado para aceder rankings específicos do paciente, caso existam.

Output

- **result : dict**

  Dicionário com os rankings dos algoritmos para cada situação.

Notas

Caso seja introduzido o id do paciente é guardado um ranking para esse paciente. Caso não seja especificado um id, é guardado o ranking para as variáveis utilizadas, usando a função save_rankings.

save_rankings(rankings, patient_id)

Não retorna qualquer valor. Quando os parâmetros são inválidos é retornada uma mensagem de erro. Caso os parâmetros sejam válidos, a função guarda os valores dos rankings num ficheiro.

Parâmetros

- **rankings : dict**

Dicionário com os rankings. Dividido por deteção de anomalias e imputação de valores em falta, cada um com os seus algoritmos e cada algoritmo com os seu rankings.

- **patiend_id : None ou int**

O parâmetro do id do paciente pode estar vazio (None) ou pode ser um número inteiro. Usado para aceder rankings específicos do paciente, caso existam.

Output

- **result : None**

Notas

Caso seja introduzido o id do paciente é guardado um ranking para esse paciente. Caso não seja especificado um id, é guardado o ranking para as variáveis utilizadas.

get_rankings(params, patient_id)

Retorna um dicionário com os rankings do algoritmos. Quando os parâmetros são inválidos é retornada uma mensagem de erro. Caso os parâmetros sejam válidos, a função retorna os valores de rankings de um determinado ficheiro.

Parâmetros

- **params : list(string)**

Lista com as variáveis do dataset, todas do tipo string.

- **patiend_id : None ou int**

O parâmetro do id do paciente pode estar vazio (None) ou pode ser um número inteiro. Usado para aceder rankings específicos do paciente, caso existam.

Output

- **result : dict**

Dicionário com os rankings. Dividido por deteção de anomalias e imputação de valores em falta, cada um com os seus algoritmos e cada algoritmo com os seu rankings.

Notas

Caso seja introduzido o id do paciente é obtido o ranking para esse paciente. Caso não seja especificado um id, é obtido o ranking para as variáveis utilizadas.

Apêndice B - Testes Realizados

Id do teste: 5

Descrição do teste: Verificar se a percentagem de anomalias respeita a percentagem que foi selecionada previamente

Passos do teste:

- Criar um dataset com anomalias
- Verificar se a percentagem de anomalias é igual ou próxima da introduzida, usando o array com a posição das anomalias

Dados usados para teste:

- Dataset completo
- Percentagens de anomalias (10, 20, 30)
- Percentagem de incremento (10)

Resultado expectável: A percentagem de anomalias é igual à passada como input

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

Id do teste: 6

Descrição do teste: Verificar se as anomalias foram criadas nos locais que foram selecionados

Passos do teste:

- Criar um dataset com anomalias
- Comparar o dataset original com o dataset com anomalias
- Determinar as posições dos valores que foram alterados
- Comparar as posições encontradas com as posições geradas

Dados usados para teste:

- Dataset completo
- Percentagens de anomalias (10)
- Percentagem de incremento (10)

Resultado expectável: As posições encontradas são todas iguais às geradas

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

Id do teste: 7

Descrição do teste: Verificar se os valores que são outliers sofreram um incremento que foi selecionado previamente

Passos do teste:

- Criar um dataset com anomalias
- Comparar os valores do dataset alterado com os valores originais
- Calcular a percentagem de incremente/redução

Dados usados para teste:

- Dataset completo
- Percentagens de anomalias (10)
- Percentagem de incremento (10, 20, 30)

Resultado expectável: As posições encontradas são todas iguais às geradas

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

Id do teste: 8

Descrição do teste: Verificar se não ficaram valores em falta por preencher após o uso de cada algoritmo de imputação

Passos do teste:

- Passar dataset incompleto por cada algoritmo
- Averiguar a existência de valores em falta

Dados usados para teste:

- Dataset incompleto

Resultado expectável: O dataset imputado não contém valores em falta

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

Id do teste: 9

Descrição do teste: Verificar se a interface retorna mensagem de erro, quando os valores de entrada são de um tipo não pretendido

Passos do teste:

- Passar um valor que não seja um dataset
- Averiguar se a situação de erro é despoletada.

Dados usados para teste:

- Uma string vazia, um valor inteiro

Resultado expectável: Mensagem de erro aparece

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

Id do teste: 10

Descrição do teste: Verificar se a interface retorna mensagem de erro, quando não se introduz um parâmetro de entrada

Passos do teste:

- Chamar a função da pipeline sem passar nenhuma variável de entrada

- Averiguar se a situação de erro é despoletada.

Dados usados para teste:

- Nenhum

Resultado expectável: Mensagem de erro aparece

Resultado atual: Igual ao expectável

Passou/Falhou: Passou

Apêndice C - Resultados

	Percentagem de Valores em falta		
Algoritmo	**10**	**20**	**30**
Imputação com Média	0.185	0.222	0.463
ARIMA	0.185	0.215	0.464
Knn	0.223	**0.207**	0.387
MICE	**0.184**	0.226	0.549
MissForest	0.280	0.473	**0.329**
VAR-IM	0.207	0.279	0.491

Tabela C.1: Resultados da imputação para dados com BP e BW

	Percentagem de Valores em falta		
Algoritmo	**10**	**20**	**30**
Imputação com Média	1.293	1.317	2.037
ARIMA	1.292	1.319	2.042
Knn	1.295	**1.287**	2.041
MICE	1.287	1.312	**2.027**
MissForest	**1.271**	1.337	2.109
VAR-IM	1.293	1.433	2.087

Tabela C.2: Resultados da imputação para dados com BP e BR

	Percentagem de Valores em falta		
Algoritmo	**10**	**20**	**30**
Imputação com Média	**0.365**	0.601	0.695
ARIMA	0.368	0.593	0.667
Knn	0.392	**0.525**	**0.582**
MICE	0.400	0.699	0.754
MissForest	0.462	0.554	0.763
VAR-IM	0.472	0.748	0.855

Tabela C.3: Resultados da imputação para dados com BP, HR e BW

	Percentagem de Valores em falta		
Algoritmo	**10**	**20**	**30**
Imputação com Média	**0.251**	0.588	0.539
ARIMA	0.254	0.582	0.538
Knn	0.294	0.733	0.609
MICE	0.262	0.621	0.539

MissForest	0.286	**0.520**	**0.517**
VAR-IM	0.283	0.583	0.518

Tabela C.4: Resultados da imputação para dados com BP, BW e BR

	Percentagem de Outliers\Incremento do outlier (%)								
Algoritmo	**5\10**	**5\20**	**5\30**	**10\10**	**10\20**	**10\30**	**15\10**	**15\20**	**15\30**
IQR	**0.976**	0.976	0.976	**0.964**	0.964	0.964	0.773	0.987	0.987
DBSCAN	0.750	**0.988**	**0.988**	0.667	**1.000**	**1.000**	0.786	**1.000**	**1.000**
Isolation Forest	0.902	0.927	0.939	0.893	0.917	0.929	**0.836**	0.921	0.947

Tabela C.5: Resultados da deteção de anomalias para dados com BP e BW

	Percentagem de Outliers\Incremento do outlier (%)								
Algoritmo	**5\10**	**5\20**	**5\30**	**10\10**	**10\20**	**10\30**	**15\10**	**15\20**	**15\30**
IQR	0.849	0.849	0.849	0.837	0.837	0.837	0.709	**0.959**	0.959
DBSCAN	0.738	**0.988**	**0.988**	0.477	0.727	**0.977**	0.549	0.535	0.660
Isolation Forest	**0.884**	0.895	0.895	**0.895**	**0.895**	0.930	**0.807**	0.946	**0.959**

Tabela C.6: Resultados da deteção de anomalias para dados com BP e BR

	Percentagem de Outliers\Incremento do outlier (%)								
Algoritmo	**5\10**	**5\20**	**5\30**	**10\10**	**10\20**	**10\30**	**15\10**	**15\20**	**15\30**
IQR	0.932	0.932	0.932	0.812	0.912	0.912	0.657	0.957	0.957
DBSCAN	**0.989**	**0.989**	**0.989**	0.700	**0.987**	**0.987**	0.636	0.721	0.971
Isolation Forest	0.875	0.886	0.898	**0.900**	0.963	0.975	**0.843**	**0.971**	**0.971**

Tabela C.7: Resultados da deteção de anomalias para dados com BP, HR e BW

	Percentagem de Outliers\Incremento do outlier (%)								
Algoritmo	**5\10**	**5\20**	**5\30**	**10\10**	**10\20**	**10\30**	**15\10**	**15\20**	**15\30**
IQR	0.927	0.927	0.927	0.825	0.925	0.925	**0.745**	**0.870**	0.932
DBSCAN	0.625	**0.988**	**0.988**	0.600	0.887	0.887	0.562	0.799	0.799
Isolation Forest	**0.939**	0.939	0.963	**0.950**	**0.963**	**0.975**	0.696	0.848	**0.973**

Tabela C.7: Resultados da deteção de anomalias para dados com BP, BW e BR