Sistema de Deteção de Surtos através do Twitter

Mestrado em Gestão de Sistemas de Informação Médica

João Filipe da Silva Cunha

Leiria, Setembro de 2019

Sistema de Deteção de Surtos através do Twitter

Mestrado em Gestão de Sistemas de Informação Médica

João Filipe da Silva Cunha

Dissertação realizada sob a orientação da Professora Doutora Catarina Silva e do Professor Doutor Mário Antunes.

Leiria, Setembro de 2019

Originalidade e Direitos de Autor

A presente dissertação é original, elaborada unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para a elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Curso de Mestrado em Gestão de Sistemas de Informação Médica, no ano letivo 2018/2019, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes

Agradecimentos

Quando se inicia um projeto com esta dimensão, muitas decisões são necessárias tomar a vários níveis. Conciliar a nossa vida profissional intensa mais a realização de um projeto é um desafio que requer um esforço adicional e emocional.

Foi bastante difícil chegar a este ponto, mas queria fundamentalmente agradecer aos meus dois professores orientadores, Doutora Catarina Silva e Doutor Mário Antunes, porque apesar deste processo difícil, longo, demorado e de todas as circunstâncias, sempre me incentivaram a concluir o projeto mostrando sempre abertura para essa realidade.

Quero também agradecer ao professor Doutor Rui Rijo pelo incentivo e por sempre conseguir mostrar várias soluções e visões para resolvermos os nossos problemas.

Por fim, agradecer a todas as pessoas que direta ou indiretamente foram importantes para me influenciar com o positivismo necessário para não desistir.

Nota Prévia

Do trabalho efetuado resultou o seguinte artigo submetido e apresentado em conferência:

- *Cunha, J., Silva, C., & Antunes, M. (2015). Health Twitter Big Data Management with Hadoop Framework. In Procedia Computer Science (Vol. 64, pp. 425–431). Elsevier. https://doi.org/10.1016/j.procs.2015.08.536*

Resumo

O presente trabalho, teve como objetivo a construção de um sistema de deteção de surtos, nomeadamente de sarampo, com dados recolhidos a partir da rede social Twitter utilizando uma plataforma Web para executar algoritmos de *Machine Learning*. Com os modelos resultantes, pretendeu-se detetar em tempo real a ocorrência de surtos de sarampo (ou outras doenças) providenciando uma ferramenta útil para fins de monitorização da saúde pública. Foram utilizados vários *datasets* de diferentes dimensões e quatro algoritmos: *LinearSVC, Multiplayer Perceptron, Random Forest* e *Logistic Regression*. À exceção do *Random Forest* todos os algoritmos conseguiram um bom desempenho na deteção de surtos de sarampo, mas constatou-se que para tal é necessário ter um *dataset* com muitos *tweets* e, sobretudo, com igual número de *tweets* em ambas as classes (surto e não surto).

Palavras-chave: Twitter, *machine learning*, deteção de surtos, *data mining*, saúde, epidemiologia.

Abstract

This work aimed to develop an outbreak detection system, for detecting measles, that uses Twitter data to spot outbreaks in real time. In order to achieve this, we implemented a Web application, to run *Machine Learning* algorithms. With the constructed model we want to identify the presence of measles outbreaks (or other diseases) so we can provide a useful tool to public health surveillance. Several datasets (with different sizes) and four training algorithms were used: *LinearSVC*, *Multiplayer Perceptron, Random Forest and Logistic Regression*. Except for *Random Forest*, every algorithm performed well in detecting measles outbreaks. It is important to have a dataset with a lot of *tweets* and, above all, with an equal number of *tweets* in both classes (outbreak and not outbreak), otherwise the algorithm will not perform acceptable.

Keywords: Twitter, *machine learning*, outbreak detection, *data mining*, health, epidemiology.

Índice

Originalidade e Direitos de Autor ... **iii**

Agradecimentos ... **iv**

Nota Prévia ... **v**

Resumo ... **vi**

Abstract ... **vii**

Lista de Figuras ... **xi**

Lista de siglas e acrónimos ... **xiv**

1. Introdução ... **1**

1.1. Motivação ... **2**

1.2. Objetivos ... **3**

1.3. Estrutura do Documento ... **3**

2. Redes Sociais em Saúde ... **5**

2.1. Introdução ... **5**

2.2. As Redes Sociais ... **7**

2.2.1. Redes sociais e *Data Mining* ... 8

2.2.2. Técnicas de *Data Mining* ... 8

2.3. O Twitter ... **12**

2.3.1. Definição e funcionamento ... 12

2.3.2. Aplicações do Twitter na área da saúde ... 13

2.4. Aplicação das redes sociais na área da saúde ... **14**

2.5. Aplicações de deteção de surtos em redes sociais ... **18**

2.6. Conclusão ... **24**

3. Tecnologias de *Machine Learning* ... **25**

3.1. Introdução ... **25**

3.2. Machine Learning ... **26**

3.3. Tipos de algoritmos em *Machine Learning* ... **29**

3.3.1. Algoritmos de aprendizagem supervisionada ... 30

3.3.2. Algoritmos de Aprendizagem não supervisionada ... 33

3.4. Redes neuronais **35**
3.4.1. Redes neuronais *feedforward* 39
3.4.2. *Backpropagation* 39
3.4.3. *Recurrent Neural Networks* (RNN) 39
3.4.4. *Long Short-Term Memory (LSTM)* 40
3.4.5. *Convolutional Neural Network* (CNN) 41
3.4.6. *Gated Recurrent Unit* (GRU) 41
3.4.7. *Random Forest* 42
3.4.8. *Multilayer Perceptron* 42

3.5. *Frameworks* de *Deep Learning* **42**
3.5.1. *TensorFlow* 42
3.5.2. *Keras* 43
3.5.3. *Apache Mahout* 44
3.5.4. *Scikit-learn* 44
3.5.5. *Apache Hadoop* 44
3.5.6. *Apache Spark* 45

3.6. O *Machine Learning* aplicado à área da saúde **45**

3.7. Conclusão **47**

4. Proposta **48**

4.1. Introdução **48**

4.2. Solução Proposta **48**

4.3. Arquitetura **49**
4.3.1. *Spring Boot framework* 50
4.3.2. *Spring Social Twitter* 51
4.3.3. Padrão *Model-View-Controller* (MVC) 51
4.3.4. *Apache Spark* 52
4.3.5. *Tomcat* 53
4.3.6. *Object-Relational Mapping (Hibernate)* 53
4.3.7. MySQL 54
4.3.8. Modelo de Dados 54
4.3.9. *Plotly* 56

4.4. Conclusão **56**

5. Implementação **57**

5.1. Introdução **57**

5.2. Estrutura do projeto **57**

5.3. Gestão de *tags*, *filtros* e *datasets* **58**

5.4. Treino dos modelos **60**

5.5. Teste dos modelos **67**

5.6. Recolha de tweets **68**

5.7. Classificação de tweets **71**

5.8. Conclusão **74**

6. Análise de Funcionamento **76**

6.1. Introdução **76**

6.2. Análise dos dados recolhidos **76**

6.3. Análise dos resultados **80**

6.4. Conclusão **85**

7. Conclusões e trabalho futuro **87**

Referências Bibliográficas **88**

Lista de Figuras

Figura 1 – Taxa de penetração das *social media* em 2019 [6]. 6

Figura 2 - Exemplo da rede social *Zachary's Karate Club* em forma de grafo [12]. 10

Figura 3 - Média mensal de utilizadores ativos no Twitter em milhões [17]. 12

Figura 4 - Exemplo de um *tweet*..... 13

Figura 5 - Ilustração das fases do processo de DM [70]. 27

Figura 6 - Ilustração do fluxo de aprendizagem por reforço [76]. 30

Figura 7 - Espaço euclidiano representativo das dimensões de um tipo de flor (largura da pétala, comprimento da pétala e comprimento da sépala [73]. 31

Figura 8 - Ilustração do conceito de aprendizagem baseada em múltiplas instâncias [75]. 32

Figura 9 - Representação de um espaço com duas classes utilizando o SVM [78]. 33

Figura 10 - Ilustração do processo do método *dividir para conquistar* [79]..... 35

Figura 11 - Ilustração de neurónio biológico e neurónio artificial [81]. 37

Figura 12 - Rede de nós com três camadas [80]. 37

Figura 13 - Exemplo de rede *feedforward* [82]. 39

Figura 14 - Exemplo do comportamento de uma RNN [83]. 40

Figura 15 - Exemplo do fluxo de dados num neurónio com memória [80]. 41

Figura 16 - Diagrama arquitetural do Sistema de deteção de surtos. 49

Figura 17 - Exemplo de um ficheiro *pom* [110]. 51

Figura 18 - Exemplo do fluxo do padrão MVC [111]. 51

Figura 19 - Algoritmos disponibilizados pela biblioteca do *Spark* [115]..... 53

Figura 20 - Diagrama do modelo de dados do Sistema de deteção de surtos..... 55

Figura 21 - Exemplo de um ficheiro de configuração *yaml*. 58

Figura 22 - Página *Manage Tags* onde é possível criar novas *tags* ou pesquisar por uma já existente. 58

Figura 23 - Método *create()* da classe *TagController*. 59

Figura 24 - Página *Manage Filters*. 59

Figura 25 - *Home page* da aplicação. 60

Figura 26 - Página *Classification* para efetuar o treino dos classificadores a partir do *dataset*. 61

Figura 27 - Tabela e gráfico com resultados do treino. 61

Figura 28 - Implementação do método *trainDataset()* que devolve a vista com os resultados do treino. 62

Figura 29 - Implementação do método *trainClassifiers()*. 64

Figura 30 - Implementação do método *runPythonScript()*. 65

Figura 31 - Implementação do objeto *TrainingResult*. 66

Figura 32 - Resultados do treino de classificadores para um *dataset*. 66

Figura 33 - Conjunto de opções de visualização do gráfico. 67

Figura 34 - Gráfico com resultados do teste. 67

Figura 35 - Implementação do método testDataset() que devolve a vista com os resultados do teste. 68

Figura 36 - Resultados das métricas dos modelos com a opção para classificação de *tweets*. 68

Figura 37 - Página *Collector*. 69

Figura 38 - Lista de *tweets* obtida a partir da API do Twitter. 69

Figura 39 - Implementação do método *searchByHashTag()* da classe *TwitterClient*. 70

Figura 40 - Implementação da classe *Tweet* com os seus parâmetros. 70

Figura 41 - Lista de *tweets* com os resultados da classificação para cada *tweet*. 71

Figura 42 - Histograma com o número de *tweets* classificados com 0 e 1 para cada modelo. 72

Figura 43 - Excerto do script *classify.py* em *python* utilizado para classificar os *tweets* com base nos modelos treinados. 72

Figura 44 - Método *collect* da classe *CollectorController*. 73

Figura 45 - Implementação do método *classify()*. 73

Figura 46 - Exemplo de um ficheiro HTML utilizado para mostrar os resultados da classificação 74

Figura 47 - Utilização do método *CountVectorizer()* para vectorização dos dados. 77

Figura 48 - Tabela com as métricas utilizadas neste estudo. 79

Figura 49 - Resultados do treino para o primeiro *dataset*. 80

Figura 50 - Resultados do treino de modelos. 81

Figura 51 - Resultados da Classificação do *dataset* de teste. 84

Figura 52 - Gráfico comparativo do desempenho dos diferentes algoritmos. 84

Lista de siglas e acrónimos

ATAM	Ailment Topic Aspect Model
ASIC	Application Specific Integrated Circuits
CDC	Centers for Disease Control and Prevention
CNN	Convolutional Neural Network
CPU	Computer Process Unit
CRF	Conditional Random Field
CRISP-DM	Cross Industry Standard Process for Data Minning
DAG	Direct Acylic Graph
DM	Data Mining
FN	Falsos Negativos
FP	Falsos Positivos
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HDFS	Hadoop Distributed File System
iPROs	Invisible Patient Reported Outcomes
JDBC	Java Database Connectiviy
JPA	Java Persistence API
JSP	JavaServer Pages
LDA	Latent Dirichlet Allocation
LSTM	Long Short-Term Memory
ML	Machine Learning
MVC	Model-View-Controller
ORM	Object-Relational Mapper
POO	Programação Orientada a Objetos
PLN	Processamento de Linguagem Natural
PSPT	Perturbação de Stress Pós-Traumátrico
RFR	Random Forest Regression
RLM	Regressão Linear Múltipla
RLS	Regressão Linear Simples
RNN	Recurrent Neural Networks

SNC	Sistema Nervoso Central
SVM	Support Vector Machine
THS	Twitter Health Surveillance
VIH	Vírus da Imunodeficiência Humana
VN	Verdadeiros Negativos
VP	Verdadeiros Positivos

1. Introdução

Nos últimos anos, não só o volume de dados gerados na Internet tem crescido significativamente, como as constantes inovações no campo das TIC (*Tecnologias de Informação e Comunicação*) têm possibilitado novas oportunidades de serviços capazes de lidar com essa grande quantidade de dados.

Atualmente, as pessoas encontram-se constantemente ligadas às redes sociais partilhando detalhes sobre o seu quotidiano e tornando estas tecnologias parte integrante da sua vida pelo que as redes sociais constituem uma forma privilegiada de partilha de informação [4]. As redes sociais são uma forma fácil e rápida de partilhar informação gerando uma grande quantidade de dados o que tem impactos relativamente à arquitetura de armazenamento de dados [9]. Assim, a análise dos dados proveniente das redes sociais surge com preponderância e como uma oportunidade na busca de informações relevantes em diversas aplicações. Quando explorados e analisados, estes dados podem ser uma importante fonte de rentabilidade para organizações públicas e privadas, seja através de novas formas de interação com cidadãos e clientes, seja no desenvolvimento de novos produtos, serviços ou estratégias [7][13].

Um outro foco é a construção de processos e ferramentas que permitam analisar o sentimento presente em determinada área, em relação a uma marca a um serviço ou a qualquer entidade [13]. Cada vez mais, diversas organizações utilizam este meio para obter um acesso direto às opiniões das pessoas, permitindo assim à organização, aferir a sua presença no mundo digital, identificando diferentes oportunidades para melhorar ou diversificar produtos, serviços ou estratégias.

Embora mais focadas numa perspetiva de negócio, empresas como a Google[1], a Amazon[2] ou o Ebay[3] já se baseiam em tecnologias que tentam percecionar os comportamentos das pessoas [103] [104].

[1] https://www.google.com
[2] https://www.amazon.com
[3] https://www.ebay.com

Uma área em que estas tecnologias são também usadas, que é o foco deste trabalho, é a área da saúde. Os objetivos podem ser diversos, incluindo por exemplo a melhoria do acesso à saúde, bem como a partilha de dados entre a sociedade e os diversos sistemas e/ou organizações de contexto médico e hospitalar.

Contudo, o volume de dados gerados digitalmente dificulta muitas vezes a possibilidade de uma análise precisa, visto que, tais dados não se encontram organizados nem são estruturados, ou seja, podem ser vídeos, fotografias, textos, etc.

1.1. Motivação

A área da saúde é um exemplo onde a geração de dados em redes sociais e na Internet em geral é efetuada em grande escala. Instituições públicas e privadas, marcas e cidadãos geram diariamente muitos dados que são maioritariamente dispersos e difíceis de operacionalizar de uma forma sistemática quer para cidadãos, quer para pessoal médico, quer para instituições. No entanto, existe um grande potencial de aplicação desta informação, podendo também estimular de forma natural o surgimento de redes de apoio e/ou comunidades de partilha de informação entre as pessoas com interesses comuns na área da saúde.

Neste contexto existem alguns projetos com objetivos específicos, como por exemplo, redes sociais exclusivas a médicos, como: a Doximity[4], a Doctors Way[5] ou a iMeds[6]. Outros exemplos, são soluções simples como a criação de grupos dedicados nas redes sociais mais populares como o Facebook[7] e o Twitter[8], mas não existem ainda soluções direcionadas que permitam tirar total partido desta informação.

Verifica-se assim, a necessidade de ferramentas que permitam o tratamento e/ou filtragem desses dados públicos, como por exemplo, a análise de sentimento dos utilizadores em relação a um determinado serviço, tópico ou produto na área da saúde, sendo essa a área em que se enquadra este trabalho.

A ideia de utilização de dados abertos (*open source*) de uma forma inteligente tem vindo a afirmar-se numa área denominada *Open Source Intelligence (OSINT)* com o objetivo de

[4] https://www.doximity.com
[5] https://doctorsway.com
[6] https://imeds.co.za
[7] https://facebook.com
[8] https://twitter.com

obter informação ou detetar padrões em dados disponíveis para todos de forma a acrescentar informação valiosa [1]. A área da saúde, dado o seu interesse generalizado de uma forma global, e a necessidade de acesso a informação, será sem dúvida uma das aplicações óbvias deste tipo de abordagens. Note-se, no entanto, que, sendo uma área sensível, qualquer solução deve ser o mais aberta possível suportando decisões de uma forma interpretável.

Assim, ter uma solução *web* desse tipo e que seja disponível de forma gratuita, seria uma colaboração na prevenção de epidemias, usando as mensagens escritas nas redes sociais para detetar um possível surto.

1.2. Objetivos

Apresentam-se de seguida os objetivos gerais definidos para este trabalho:

- Analisar o estado da arte de sistemas de recolha de informação pública na área da saúde;
- Analisar o estado da arte das tecnologias para obter informação das redes sociais e da forma de processamento.
- Construir um conjunto de dados (*dataset*) que permita simular um grande volume de informação disponível na rede social Twitter.
- Desenhar uma arquitetura para a construção de modelos de aprendizagem e deteção de padrões com base no conjunto de dados construído.
- Implementar e testar a arquitetura proposta e os seus modelos no conjunto de dados.
- Desenvolver um protótipo de aplicação que permita a utilização dos modelos em dados reais de uma rede social em tempo real.

1.3. Estrutura do Documento

O presente documento encontra-se estruturado da seguinte forma: Capítulo 2 *Redes Sociais Em Saúde*; Capítulo 3 *Tecnologias de Machine Learning*; Capítulo 4 *Proposta*; Capítulo 5 *Implementação*; Capítulo 6 *Análise de Funcionamento* e Capítulo 7 *Conclusões e Trabalho Futuro*.

No Capítulo 2 é efetuada uma revisão teórica sobre o tema *Redes Sociais em Saúde*. Primeiramente, é definido o conceito de rede social e de que forma é que as redes socias podem ser utilizadas como fonte de dados em processos de *Data Mining* bem como algumas técnicas de *Data Mining* utilizadas neste campo. Seguidamente é abordado o caso particular do Twitter e procura-se mostrar a importância do seu estudo em processos de *Data Mining*

e vários exemplos de trabalhos já realizados utilizando esta rede social. O capítulo termina com algumas considerações sobre os desafios nesta área.

O Capítulo 3 constitui uma revisão da literatura sobre o conceito de *Machine Learning*. Nele, é apresentada uma definição mais exaustiva deste conceito e de como ele se insere no processo de *Data Mining* tendo em conta os métodos utilizados neste campo. É também apresentada uma descrição sobre os algoritmos de aprendizagem mais comuns incluindo aqueles utilizados neste projeto. O conceito de redes neuronais é também profundamente abordado e são apresentados alguns exemplos de utilização destes algoritmos na área da saúde bem como algumas *frameworks* de *deep learning* que podem auxiliar o trabalho nesta área.

O Capítulo 4 apresenta a proposta de projeto e descreve os objetivos do projeto, a arquitetura da aplicação desenvolvida, as tecnologias utilizadas e o modelo de dados proposto.

O Capítulo 5 versa sobre as questões de implementação da aplicação *Sistema de Deteção de Surtos*. Descreve a estrutura do projeto e as diferentes funcionalidades disponibilizadas bem como a forma como foram implementadas e como podem ser utilizadas de forma a treinar modelos e a classificar *tweets* tendo por base os modelos treinados.

O Capítulo 6 é dedicado à análise de funcionamento e apresenta uma discussão dos resultados alcançados em classificação tendo em conta os diferentes algoritmos de treino e as diferentes métricas analisadas.

O Capítulo 7 é o capítulo final onde são apresentadas as conclusões da realização deste trabalho, os obstáculos encontrados e as sugestões de trabalhos a desenvolver no futuro.

2. Redes Sociais em Saúde

2.1. Introdução

Apesar de estarmos na iminência da era da web 4.0, foi na web 2.0, em 2004, que apareceram os serviços partilhados por todos os utilizadores gerando conteúdo de forma colaborativa e não apenas numa ótica de consumo de informação [7][53]. De facto, a Internet e, em particular, as redes sociais têm assumido um papel muito relevante na publicação de eventos, notícias e até na expressão de sentimentos. Assim, as redes sociais têm-se tornado objetos de grande interesse por permitirem uma análise em tempo real e a elaboração de predições de forma mais precoce do que outros meios. Este tipo de abordagem é utilizado nas mais diversas áreas, tais como: a monitorização do trânsito, a área da gestão, previsão de desastres, as notícias, entre outros. [53].

Esta partilha de informação estendeu-se, também, à área dos cuidados de saúde [3]. Desta forma, foram-se criando relações e estabelecendo contactos, fazendo com que se acentuasse o crescimento do *social media*, onde se destacam as redes sociais. Atualmente as redes sociais fazem parte do quotidiano das pessoas tornando-se as principais ferramentas para o aumento da informação gerada na Internet [4]. Esta massificação e intensificação do uso das redes sociais acarretou mudanças no comportamento das pessoas e no fluxo de informação da sociedade, facilitando a sua circulação. Podendo, por exemplo, tornar-se numa dependência e de uma necessidade de estar sempre informado sobre acontecimentos.

As redes sociais constituem espaços privilegiados de interação social, comunicação e partilha de informação providenciando informação de forma rápida e constante. Uma vez que a informação já não se encontra limitada a uma localização física a sua difusão é, agora, muito maior pelo que as redes sociais contribuem largamente para a construção, partilha e difusão de informação [4].

Em 2018, estimou-se cerca de 2,65 mil milhões de utilizadores de redes sociais por todo o mundo número que tenderá a aumentar para cerca de 3,1 mil milhões até 2021 [5]. O gráfico da Figura 1 permite ter uma ideia desta evolução ao longo do tempo.

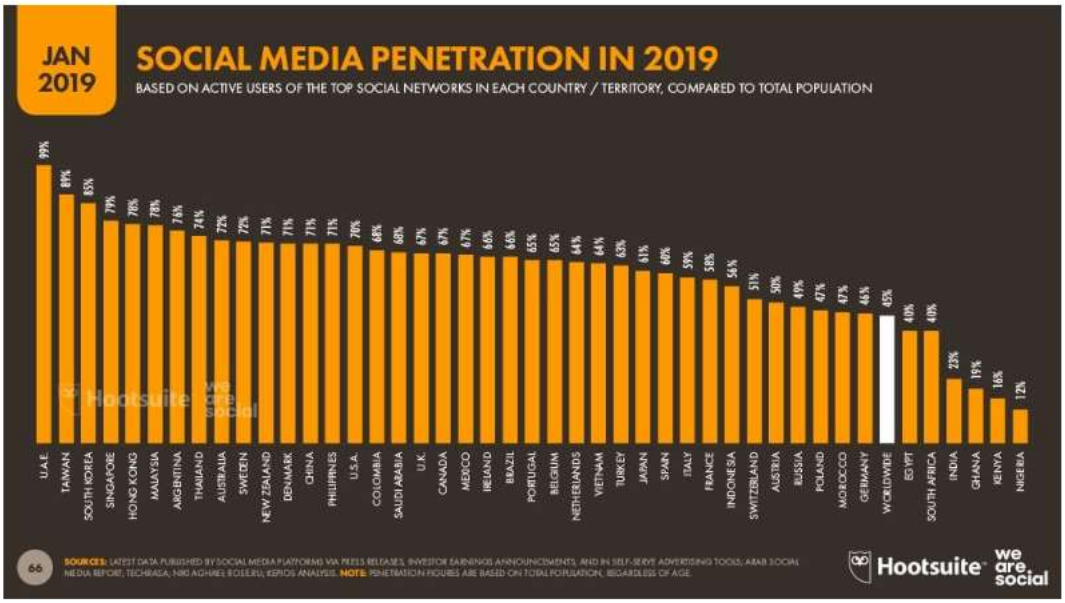

Figura 1 – Taxa de penetração das *social media* em 2019 [6].

A nível aplicacional alguns dos *websites de social media* mais utilizados são o Youtube[9], o Facebook e o Twitter [7]. No âmbito deste projeto, focamo-nos, em particular, nas redes sociais que descreveremos na secção *2.2*.

O grande crescimento na diversidade de ferramentas de *social media* e massiva utilização destas formas de comunicação e de partilha têm-nas tornado interessantes na investigação sobretudo ao nível da análise dos seus conteúdos utilizando processos de inteligência artificial.

O presente projeto visa a elaboração de um sistema que permita detetar surtos de doenças através do Twitter pelo que iremos focar-nos nesta rede social em particular. Assim, este capítulo encontra-se estruturado da seguinte forma: a secção *2.2* apresenta uma breve explicação sobre a utilidade, técnicas e métodos utilizados no processo de *data mining* para trabalhar dados provenientes das redes sociais. Nas secção *2.3* é elaborada uma breve descrição do Twitter e do seu funcionamento seguida de uma explicação sobre como pode ser utilizado na área da saúde, a secção *2.4* descreve como é que as redes socias podem ser utilizadas para fornecer informação relevante para a área da saúde e a secção *2.5* foca-se em mostrar como é possível utilizar as redes sociais para identificar surtos de doenças de forma

[9] https://youtube.com

rápida bem como os desafios e limitações inerentes. Por fim, a secção e *2.6* apresenta um resumo sobre as ilações mais relevantes deste capítulo.

2.2. As Redes Sociais

As redes sociais podem ser definidas como um tipo de aplicação computacional que promove relações sociais entre pessoas partindo de interesses generalistas, conhecimentos e atividades [8]. Nos últimos anos tem crescido a investigação sobre as redes sociais ao nível da análise dos conteúdos nelas partilhados. Estas investigações procuram métodos, técnicas e modelos que permitam tirar partido daquilo que estas redes disponibilizam: grandes quantidades de dados a partir dos quais se pretende gerar informação. Por exemplo, no âmbito da saúde pública, o uso das redes sociais pode revelar-se rico em fornecer indícios precoces sobre epidemias e dar o feedback sobre as políticas de prevenção e resposta implementadas [7]. Ao nível da análise do *social media*, a abordagem passa por desenvolver ferramentas que recolhem, analisam, resumem e disponibilizam dados a partir, por exemplo, das redes sociais. A partir destes dados, numa perspetiva de inteligência, pretende-se derivar informação utilizável e desenvolver procedimentos de tomada de decisão projetando novas arquiteturas e *frameworks* que permitam beneficiar do conhecimento adquirido [7]. No entanto, é preciso ter em conta que as redes sociais são enormes e a recolha de informação do mundo real a partir destas estruturas pode ser desafiante ao nível, por exemplo, dos processos de extração e processamento dessa informação bem como ao nível da representação e armazenamento da mesma [9]. Na secção *2.5.1* podemos ter uma melhor perceção dos desafios e limitações inerentes a este tipo de abordagens.

Utilizar as redes sociais para criar mecanismos de decisão relacionados com a saúde pública parece ser de grande interesse para os profissionais de saúde [13]. As pessoas colocam *posts* nas redes sociais descrevendo as suas vivências em relação a sintomas e tratamentos relacionados com o seu estado de saúde. Este tipo de *posts* são denominados *Invisible Patient Reported Outcomes* (iPRO) uma vez que fornecem informação clínica relevante, embora de difícil extração utilizando os métodos tradicionais.

Assim, a utilização de classificadores do conteúdo dos *posts* pode ajudar a detetar indivíduos que publicam informações sobre as suas doenças. Ao reunir estas publicações sobre os *posts* é possível efetuar uma análise de sentimento para encontrar temas emocionais semelhantes entre os dados, como por exemplo através de expressões ou carateres expressivos. Desta

forma os iPROs recolhidos das redes sociais podem fornecer informação relevante para os profissionais de saúde e até para os organismos públicos da área da saúde [13]. Em última instância, o objetivo destes procedimentos é o de promover e proteger as comunidades de saúde utilizando informação proveniente das redes sociais [14].

Como o âmbito deste trabalho incide sobre o Twitter, iremos focar-nos particularmente em trabalhos realizados com esta plataforma (para mais informação sobre o Twitter, consultar a secção *2.3*).

2.2.1. Redes sociais e *Data Mining*

Dada a riqueza e a quantidade de informação que circula nas redes sociais, estas fornecem material importante para ser usado em técnicas de *Data Mining* (DM). Estas técnicas envolvem a descoberta de modelos preditivos a partir de métodos, gerando a interpretação imparcial dos dados que pode ser usada em muitas áreas para gerar conhecimento ou prever padrões de comportamento [10]. Quanto ao seu propósito, o *DM* pode ser categorizado de duas formas: *DM preditivo* e *DM descritivo*. Enquanto o primeiro se foca em criar modelos a partir dos dados recolhidos, o segundo encarrega-se de gerar data sets a partir dos dados [10]. O processo de DM será descrito de forma mais detalhada no capítulo *3.2*.

2.2.2. Técnicas de *Data Mining*

Existem vários tipos de análise de redes sociais que podem ser distribuídos por duas categorias principais: análise de redes sociocêntrica e análise de redes egocêntrica. A análise de redes sociocêntrica surgiu da sociologia e pretende quantificar a interação entre um grupo de pessoas com o objetivo de detetar padrões estruturais globais sendo a mais utilizada em análise de redes socias. A análise egocêntrica emergiu da psicologia e da antropologia e pretende quantificar as interações dentro de um indivíduo de forma a generalizar as características encontradas em redes individuais [9].

As técnicas aplicadas para recolha de dados e geração de modelos são diversificadas e incluem: *caracterização, classificação, regressão, associação, deteção de desvios, análise de links, clustering, deteção de alterações e procura de padrões sequenciais.* [10].

A *caracterização* visa resumir e generalizar as diferentes características dos dados. A *classificação* distribui os dados por diferentes classes. A *regressão* opera da mesma forma que a *classificação*, mas prevê objetos contínuos, contrariamente à classificação que trabalha com dados discretos. A *associação* procura associações entre várias bases de dados e

também entre os atributos de uma base de dados. A *deteção de desvios* procura desvios significativos entre o valor esperado do objeto e o seu valor atual. A análise de *links* procura criar modelos a partir de padrões relacionais detetados através das ligações entre os objetos. O *clustering* agrupa dados em várias classes capazes de descrever esses dados conseguindo formar pequenos grupos a partir de grandes *datasets* sendo o seu foco o que conseguir aumentar a semelhança entre os objetos da mesma classe e reduzir a semelhança entre as classes. A *procura de padrões sequenciais* visa encontrar padrões que ocorrem frequentemente nos dados [10].

No caso das redes sociais, o DM pode ser aplicado usando vários tipos de métodos, dos quais se destacam os métodos baseados em grafos (teorias que estudam as relações entre os objetos de um conjunto), extração de texto (*text mining),* técnicas de *machine learning*, modelos de tópicos, entre outros. Não obstante, muitas vezes estes métodos interligam-se para dar origem a novos modelos.

Os métodos de grafos, como a classificação, deteção, medição de padrões, predição, processamento de dados, evolução e estrutura dos dados, modelagem e comunidades dos dados permitem responder a questões nas mais diversas áreas de forma mais rápida [10]. Isto é possível através da construção de modelos preditivos, com base num conjunto de dados fornecido à partida, que vai ser analisado tentando perceber de que forma os dados se relacionam e também permitem extrair relações de causa-efeito entre dados de input e output. Após uma fase de aprendizagem, ao alimentar esses modelos com novos dados, é possível responder a perguntas de forma automática. Por exemplo, na área da saúde, utilizam-se dados provenientes de diversas fontes para construir modelos capazes de identificar a predominância de doenças numa dada população, saber quando determinadas epidemias ocorrem e quais os fatores associados a elas ou até descobrir como sintetizar novos medicamentos através de análise exploratória.

O estudo das redes socias é uma área multidisciplinar que surgiu do cruzamento de várias disciplinas, como: a sociologia, a estatística, a psicologia e a teoria dos grafos. De acordo com a teoria dos grafos, as redes sociais consideram que nas relações socias entre os indivíduos estes são representados por pontos/nós e as relações são representadas por linhas/*links* [12]. A Figura 2 mostra um exemplo da representação de uma rede social de acordo com a teoria dos grafos.

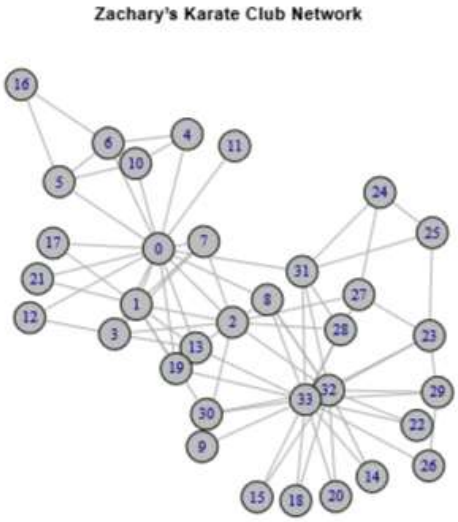

Figura 2 - Exemplo da rede social *Zachary's Karate Club* em forma de grafo [12].

Na literatura encontram-se vários exemplos de como a elaboração de grafos pode contribuir para a deteção de comunidades afetadas pela gripe [67]. Este é um exemplo de um trabalho que utilizou o método da extração de texto, grafos e também métodos estruturados de DM [67].

Muita da informação retirada das redes sociais encontra-se sob a forma de texto pelo que são necessárias metodologias específicas para identificar informação textual. Por exemplo, no caso concreto de aplicação das redes sociais à área da saúde, os métodos baseados em aprendizagem supervisionada, a extração de texto (através de palavras-chave, por exemplo), análise de sentimento, entre outros [55]. A extração de texto é um processo que utiliza dados não estruturados, para encontrar informação pertinente que é, por exemplo, um método muito usado para detetar surtos de gripe [55][67]. Ainda no âmbito da extração de texto, existem diferentes tipos de análise, entre as quais: a *análise de coocorrências* e *análise histórica de padrões*. A *análise de coocorrências* procura encontrar as frequências de determinadas palavras-chave num documento com o objetivo de encontrar publicações nas redes sociais com melhor precisão e maior relevância [68]. Já na *análise histórica de padrões* é possível usar padrões presentes em eventos históricos que ocorreram no passado para tentar prever eventos futuros. Alguns desses padrões encontram-se, por exemplo, relacionados com pesquisas efetuadas nos motores de busca ou publicações deixadas nas redes sociais [55].

Os métodos baseados em palavras-chave necessitam de um dicionário que tenha palavras relacionadas com doenças para que um texto proveniente das redes sociais seja considerado *relacionado* se contiver uma dessas palavras, ou classificado como *não-relacionado*, em caso contrário. Vários estudos sobre palavras-chave relacionadas com a gripe procuram identificar, não apenas os surtos de *influenza*, mas também doenças que se assemelham a ela usando dados provenientes do Twitter [41]. Já outros, conseguiram identificar uma relação entre as pesquisas para *influenza* e os resultados de pesquisa para palavras-chave *influenza* e *gripe* no motor de busca do Yahoo[10] [43]. Os métodos baseados em aprendizagem supervisionada também são usados para a extração de texto. Estes consistem em usar dados de treino categorizados (por humanos) para classificar a informação [41]. Por exemplo, Collier e Doan propuseram um algoritmo para detetar *tweets* relacionados com doenças usando classificadores de *Bayes* e *Support Vector Machines* (SVMs) também usados noutros estudos para treinar classificadores de forma a identificar *tweets* relacionados com a gripe [44].

Das técnicas de *machine learning* mais utilizadas destacam-se: o SVM, que é um modelo de aprendizagem supervisionada e que é dos mais usados para classificar publicações relacionadas com a gripe [2][55]; as redes neuronais, já utilizadas para rastrear a gripe numa população através de palavras-chave relacionadas com gripe e com os seus sintomas (como as redes neuronais são uma das metodologias usadas neste trabalho, falaremos delas em maior pormenor no Capítulo 3); ou o *Naive Bayes*, um classificador já utilizado por vários investigadores para analisar informação preveniente de *tweets* [69].

Por fim, os modelos de tópicos são uma abordagem em que se utilizam os dados recolhidos das redes sociais para gerar modelos que permitam identificar um determinado assunto. A título de exemplo destaca-se o *Ailment Topic Aspect Model* (ATAM) que é um modelo capaz de associar palavras com os seus tópicos subjacentes e que foi utilizado para encontrar *posts* relacionados com o tópico *doença* no Twitter [29]. Este modelo era capaz de relacionar sintomas e tratamentos com uma determinada doença.

[10] https:// https://www.yahoo.com

2.3. O Twitter

2.3.1. Definição e funcionamento

O Twitter, lançado em julho de 2006 nos Estados Unidos da América, com o intuito de funcionar como um serviço de mensagens na Internet, é uma rede social gratuita de microblogging. É uma das redes sociais mais usadas estimando-se uma média mensal de 326 milhões de utilizadores. A Figura 3 contém um gráfico com a utilização mensal média de utilizadores no Twitter e mostra a evolução da sua utilização entre 2017 e 2018.

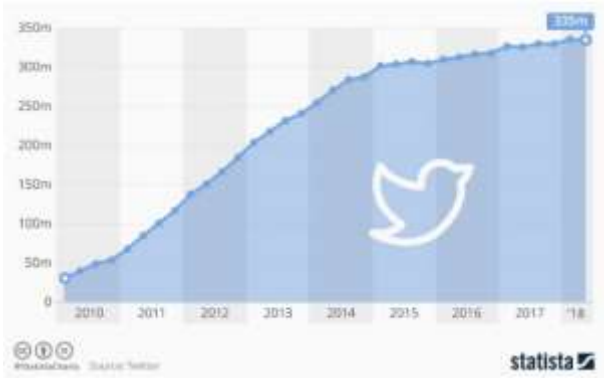

Figura 3 - Média mensal de utilizadores ativos no Twitter em milhões [17].

O Twitter proporciona a interação do indivíduo com a população onde estão incluídas entidades políticas, organizações e até celebridades [13]. Nesta rede social, os utilizadores podem partilhar diversas informações através de mensagens curtas até 280 caracteres (originalmente 140), denominadas por *tweets* [15]. Contrariamente ao Facebook que utiliza um sistema de amizade, onde as pessoas se aceitam mutuamente, no Twitter o indivíduo pode escolher livremente quem quer seguir, através da opção *follow*, e com quem pretende comunicar sem ser necessário consentimento da pessoa/grupo seguida(o) [15].

Os *tweets* podem conter no seu conteúdo um ou mais cardinais (#) antes de uma determinada palavra, de modo a demarcar o seu assunto. O uso desse caracter foi transportado também para outras redes sociais e é designado como *hashtag*.

A *hashtag* ajuda a categorizar os *tweets* com base no contexto em que está inserido, permitindo ter melhores resultados de pesquisa constituindo, assim, um categorizador de conteúdos eficaz [16]. Podem ser inseridas em qualquer parte do *tweet* e quando os utilizadores clicam nela, são redirecionados para a categoria que agrupa todos os tweets dos utilizadores que têm o mesmo assunto. A Figura 4 mostra um exemplo de um *tweet*.

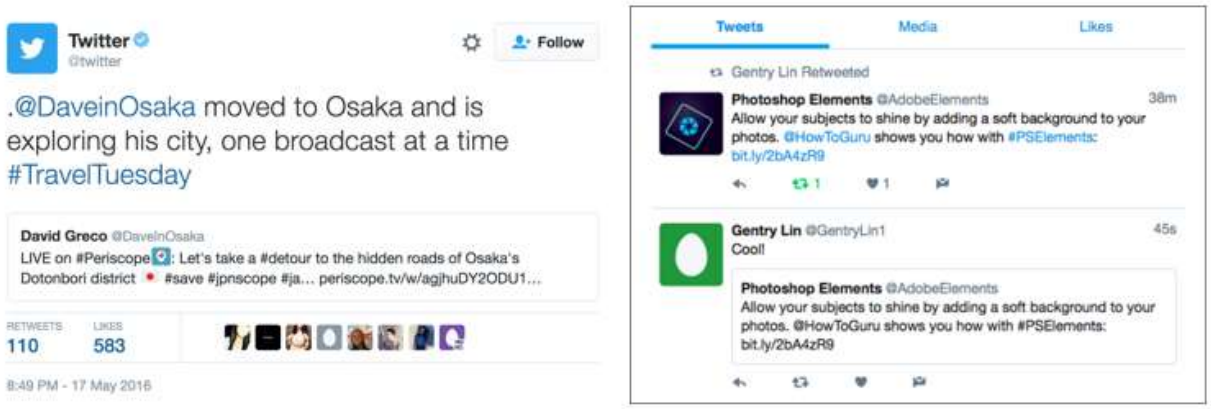

Figura 4 - Exemplo de um *tweet*.

Dada a sua popularidade, esta ferramenta tornou-se uma fonte importante de dados linguísticos carregada de opiniões, sentimento e discussão [13].

2.3.2. Aplicações do Twitter na área da saúde

Não faltam trabalhos que tentam associar a utilização do Twitter à possibilidade de reunir indícios sobre doenças que afetam a população [24]. A classificação dos dados provenientes do Twitter em categorias pode ajudar a compreender a forma como os utilizadores reagem e comunicam [25]. Alguns dos trabalhos realizados neste campo usam *queries* de pesquisa efetuadas no motor de busca da Google para monitorizar os surtos de doenças [57] e muitos deles procuram desenvolver modelos preditivos que ajudem a prever quando é que essas epidemias vão surgir [58] [59].

No entanto, existe um conjunto de desvantagens associadas a esta prática pelo que se optou por usar o Twitter como fonte de dados do presente trabalho.

Como já foi mencionado, o Twitter pode ser um aliado poderoso para os profissionais de saúde e entidades de saúde pública. Assim, têm sido realizados vários trabalhos com foco sobre a utilização do Twitter como fonte de informação para a área da saúde. Além de providenciar uma grande quantidade de dados, o Twitter permite disponibilizar essa informação praticamente em tempo real e numa escala global [25]. Isto permite que epidemias e surtos sejam detetados numa fase precoce e tem a vantagem de permitir uma comunicação mais rápida entre as entidades de saúde e o público [56]. Não obstante, os *tweets* conseguem ser mais descritivos e disponíveis para o público do que uma pesquisa nos motores de busca [53]. A análise de *tweets* em vez de *queries* de pesquisa permite ainda ir

um pouco além e analisar também o ambiente de quem publicou as mensagens obtendo assim mais informação como dados demográficos e outros detalhes que poderão enriquecer o resultado [53]. Outra das vantagens, é que a utilização do Twitter é massiva e as idades dos utilizadores são muito diversificadas contribuindo para aumentar a representatividade da amostra [54]. Uma vez que as *hashtags* podem alcançar uma vasta audiência, o foco na sua análise pode ser muito rico em fornecer informação para a análise de sentimento. Desta forma, ao comparar os sentimentos dos *tweets* com *hashtags* mais frequentes é possível clarificar a perceção do público e as suas reações relativamente a eventos mundiais [13].

Vários estudos utilizaram processos de DM no Twitter para monitorizar a saúde pública e conseguiram identificar sintomas relacionados com doenças de vários tipos [19] como iremos perceber no capítulo seguinte: *2.4*.

2.4. Aplicação das redes sociais na área da saúde

De forma a melhorar a monitorização na área da saúde têm sido desenvolvidas várias plataformas e aplicações. Brownstein lançou uma ferramenta de *social media*, o *website* HealthMap[11], vocacionado para a monitorização de doenças infeciosas que extrai informação de sites de notícias, alertas do governo, testemunhos de pessoas e outra informação sobre surtos de várias doenças em todo o mundo. Esta informação é reunida num mapa global que disponibiliza informação em tempo real [46]. O mesmo autor também lançou uma aplicação para iPhone designada por Outbreaks Near Me[12] que disponibiliza o HealthMap no telemóvel. Existe, ainda, o Flu Near You[13] criado pela American Public Health Association e pela Skoll Global Threats Fund of San Francisco através do qual as pessoas reportam o seu estado de saúde semanalmente, fornecendo informação sobre potenciais surtos [46].

A Google também tem tido a sua participação neste campo e lançou o Google Flu Trends[14] que consiste num *website* que permite a comparação entre a procura de tópicos relacionados com gripe e as taxas de incidência reportadas graficamente num mapa. Este *website* é

[11] http://healthmap.org
[12] https://www.healthmap.org/outbreaksnearme
[13] https://flunearyou.org
[14] https://www.google.org/flutrends/about

considerado um importante aliado para a deteção precoce da gripe em determinadas localizações [46].

Como já foi mencionado na secção anterior, o uso do Twitter é um poderoso aliado para alcançar este objetivo e vários são os estudos nesta área que envolvem o uso desta ferramenta. Por exemplo, Lamb et al. conseguiram rastrear e monitorizar a incidência do vírus *influenza* através da combinação de dados provenientes do *website* Google Flu Trends e Health Tweets. Este estudo concluiu que o rastreamento das taxas de infeção por *influenza* era mais eficaz quando eram utilizados dados provenientes do Twitter a nível municipal e menos eficaz se fossem recolhidos dados a nível regional ou nacional [20].

Através dos seus estudos com populações vítimas de cancro utilizando redes neuronais para classificar os *tweets* relacionados com esta temática, Crannel concluiu que a utilização de iPROs pode ajudar os profissionais de saúde a elaborar planos de tratamento mais personalizados para os seus pacientes [18]. Um trabalho realizado por Clark também concluiu, também que a análise da informação proveniente das redes sociais pode ser uma forma eficaz de elaborar estratégias de monitorização da saúde pública [13]. A área das doenças mentais também tem sido objeto de estudo. Reece, por exemplo, utilizou *tweets* para criar modelos computacionais capazes de prever o aparecimento de doenças mentais tais como a Depressão e a Perturbação de Stress Pós-Traumático (PSPT) nos utilizadores do Twitter. Estes modelos foram construídos a partir da medida do afeto, estilo linguístico e contexto dos *tweets* que depois foram usados para alimentar algoritmos de aprendizagem [21]. Dentre estes estudos, parece emergir a ideia de que é possível encontrar indícios de doença física e psicológica mesmo antes de o utilizador da rede social saber que a carrega permitindo a sua deteção numa fase mais precoce [22].

Alguns estudos usaram também metadados dos *tweets* para construir algoritmos de classificação. Alguns dos metadados usados foram: o número de seguidores, a frequência de publicação, o tamanho do *username*, a longevidade da conta, o número de *retweets* e o número de respostas ou menção ao utilizador [23]. Estes estudos também têm utilizado o ciclo de atividade humana para identificar entidades que parecem orgânicas, mas, na verdade, destinam-se a propagar conteúdo promocional [23]. Estes estudos mostram como a investigação se tem interessado em identificar contas fictícias em redes sociais (que pertencem a robots).

Rodriguez-Martínez & Garzón-Alfonso consideraram que a maioria das abordagens que visa recolher dados a partir do Twitter para rastrear a incidência de doenças sobre a população, pecam por se basearem na procura por palavras-chave o que consideram que pode conduzir a resultados menos precisos pois o aparecimento desta palavra pode não estar diretamente relacionado com a presença da doença em si. Uma outra desvantagem deste tipo de abordagens está relacionada com o facto de se focarem no treino de classificadores e análise estatística, descurando a escalabilidade dos métodos usados para extrair dados das redes sociais e processá-los. Não obstante, para implementar *streaming* de dados em larga escala e ferramentas de *big data*, são necessários vários sistemas independentes: de processamento de *streaming*, *armazenamento de dados*, ferramentas de *Machine Learning* (ML) e armazenamento de *big data*. Como este tipo de solução tem muita procura torna difícil às organizações de saúde conseguirem contratá-los.

Rodriguez-Martínez & Garzón-Alfonso apresentaram, então, a *framework* Twitter Health Surveillance (THS) que descreveram, *como uma plataforma integrada que auxilia os profissionais de saúde a recolher tweets, determinar se estão relacionados a condições médicas, extrair metadados deles e criar um big data warehouse para utilizar em análises futuras de dados*. Isto levou à classificação de três tipos de serviços disponibilizados pelo THS: aquisição de dados, classificação do *tweet* e *big data warehousing* [24].

A aplicabilidade do Twitter à área da saúde pública serve diversos propósitos, entre os quais: a monitorização de doenças, a geolocalização, a deteção de surtos (que desenvolveremos na secção *2.6*), a previsão de surtos, a reação pública e o estilo de vida [25].

No campo da monitorização de doenças destacam-se várias abordagens interessantes como o ATAM, já mencionado na secção *2.3.1*, que se propõe identificar *tweets* relativos a doenças [29], ou o M-Eco, um sistema de monitorização da doença que processa dados provenientes de diversas fontes, incluindo o Twitter, a rádio, os blogs, os fóruns e programas de televisão para obter informação relevante sobre as epidemias. O M-Eco filtra palavras-chave a partir dos dados e analisa os textos de forma a definir a sua importância [28].

No que concerne à reação pública, autores como Adrover e outros procuraram identificar utilizadores do Twitter com Vírus da Imunodeficiência Humana (VIH) e perceber se os sentimentos e os tratamentos farmacológicos podiam ser detetados através do Twitter [30]. Utilizaram várias abordagens, tais como: a filtragem por palavras-chave, algoritmos computacionais, *machine learning* e *crowdsourcing*. A análise de sentimento é uma das

abordagens mais usadas no que concerne à análise da reação pública e Behera e Eluri desenvolveram um método de análise de sentimento que se propõe monitorizar a propagação das doenças de acordo com a localização e o tempo [31]. Estes autores pretendiam medir o nível de preocupação subjacente aos *tweets* a propósito de três doenças: malária, cancro e gripe suína.

A aplicabilidade do Twitter às situações de emergência e surtos também já foi alvo de várias investigações. Por exemplo, Odlum e Yoon recolheram cerca de 42000 *tweets* relacionados com o vírus ébola durante o surto em 2014. Eles utilizaram estes dados para procurar tendências na propagação da informação, perceber o conhecimento público relativo ao vírus e tentar detetar precocemente a epidemia.

Verificaram que, efetivamente, se registou um aumento na propagação da informação sobre o ébola (fatores de risco, prevenção, etc.) no Twitter alguns dias antes de ser dado o alerta oficial sobre o surto o que mostra a importância que o Twitter pode ter como sistema de alerta precoce [32]. Também Gomide e outros apresentaram uma metodologia de monitorização *quadrimensional* para rastrear epidemias de dengue no Brasil. As quatro dimensões utilizadas foram: o volume, o número de *tweets* que mencionavam a palavra dengue; o tempo, em que altura os *tweets* eram publicados; a perceção pública, isto é, o sentimento sobre a epidemia de dengue; e a localização referente aos *tweets* [33]. Mais informações sobre a utilização do caso particular do Twitter para a prevenção de surtos poderão ser consultadas na secção *2.5* deste subcapítulo.

Como foi referido anteriormente, aplicabilidade do Twitter também é válida para efeitos de predição. Kautz e Sadilek, por exemplo, sugeriram um modelo preditivo do estado de saúde futuro de um indivíduo com 91% de precisão. Neste trabalho, milhões de *tweets* foram recolhidos e os utilizadores foram investigados através de processos de DM sobre a sua comunicação online, a localização, o ambiente e status social dos utilizadores para descrever o comportamento do utilizador. Os autores conseguiram correlacionar uma saúde fraca com a proximidade de fontes de poluição e o elevado status social com uma saúde melhor [34].

Relativamente ao estilo de vida público, alguns estudos têm indicado a importância do Twitter como meio de classificação. Assim, existem trabalhos que propuseram a modelação de dados do Twitter a partir do método *Latent Dirichlet Allocation* (LDA) para análise de tópicos de conversação no âmbito da saúde pública [25].

No que concerne à geolocalização existem algumas investigações sobre como o Twitter pode ser aplicado. Desde trabalhos que propuseram sistemas para determinar a localização geográfica dos *tweets*, como o Carmen [35] à utilização de sistemas de *geotagging* que permitem extrair informação geográfica (metadados) acerca dos *tweets*, como o *LIW-META* [36]. Este sistema utiliza informação implícita (termos relacionados com localizações) e explícita (palavras que indicam localizações) e combinam-na com informação do perfil de utilizador para inferir a geolocalização dos *tweets*, algo que este estudo conseguiu alcançar com precisão [36].

Vale a pensa referir que muitas das investigações atrás referidas centram-se em recolher dados a partir dos *tweets* ignorando os dados sobre quem os publicou, o que pode acarretar problemas de precisão e credibilidade dos resultados. No entanto, existem algumas aplicações de *machine learning* que procuram eliminar esta limitação fornecendo abordagens que têm em conta o perfil do utilizador no Twitter e procuram inseri-lo numa classificação. É o caso do estudo realizado por Pennacchiotti e Popescu onde foram utilizadas informações sobre o comportamento do utilizador, a linguagem utilizada e a estrutura da rede para inferir a orientação política e a etnia dos mesmos. Os resultados sugeriram que características associadas à linguagem por parte do utilizador mostram-se muito importantes para fornecer indícios sobre a orientação política, a identificação com uma etnia ou o interesse por um determinado assunto, pelo que esta abordagem parece ser útil para a classificação dos utilizadores [60].

Um outro estudo utilizou os dados provenientes de redes sociais, incluindo o Twitter, para compreender as opiniões e comportamentos das pessoas sobre questões relacionadas com a vacina da *influenza*. Foram implementados vários classificadores de linguagem natural para encontrar padrões comportamentais relacionados com a vacinação contra a *influenza* e foram utilizadas três dimensões: o tempo, localização geográfica e a demografia (o género). Os resultados conseguiram uma correlação elevada [64].

2.5. Aplicações de deteção de surtos em redes sociais

A epidemiologia é uma ciência dedicada ao estudo dos fenómenos de saúde, doença e da forma como estes se distribuem pela população. Esta ciência tem apostado na investigação para identificar a incidência, a distribuição e a etiologia das doenças de forma a facilitar a sua prevenção, controlo e tratamento [25]. Este é o grande objetivo dos profissionais de

saúde que procuram prevenir a doença recorrendo à monitorização da saúde pública recolhendo, gerindo, analisando e interpretando os dados de forma sistemática. Os dados são disseminados em programas que facilitam ações em saúde pública [25]. Ao mesmo tempo, os surtos/epidemias têm-se tornado cada vez mais frequentes e diversificadas o que tem contribuído para o desenvolvimento de novas ferramentas de deteção de surtos [26]. Entre estas, encontram-se as ferramentas de análise de dados digitais. Na verdade, um dos fatores mais importantes na deteção de epidemias está relacionado com a rapidez de resposta de forma a combater eficazmente a emergência de doenças infeciosas [25]. A inovação tecnológica poderá, assim, ter um importante contributo para a previsão de determinadas doenças, contributo este que poderá partir de rotinas de monitorização já existentes, como as utilizadas no caso do vírus *influenza* [27].

De entre várias redes socias, destaca-se a utilização do Twitter para a deteção de surtos possibilitando a recolha de informação em tempo real [25].

Woo e outros, realizaram um estudo com o objetivo de identificar palavras-chave coreanas para detetar epidemias de *influenza* a partir de dados recolhidos do Twitter e de *blogs*. No seu trabalho, procederam à seleção de palavras-chave, elaboração de uma série temporal de palavras-chave através de pré-processamento, construção e validação de modelos, SVM e *Random Forest Regression* (RFR). Utilizando estes métodos, conseguiram detetar epidemias de *influenza* a partir de um conjunto de 15 palavras-chave [2]. Um outro estudo realizado com dados provenientes do Twitter teve como objeto os surtos de ébola e de Zika de 2014 e providenciou vetores de palavras-chave relacionadas com estas doenças para identificar *tweets* relacionados com elas [40]. Os resultados deste estudo indicaram que os conjuntos de palavras específicas destas doenças retiradas do Twitter conseguiam extrair melhor relações semânticas significativas do que a utilização os modelos *Word2Vec* ou *Glover*.

A maioria das abordagens para identificar epidemias são *top-down* e consomem muito tempo pois é baseada em informação já conhecida como o nome da doença ou os seus sintomas. Como as abordagens *top-down* carecem de uma formalização que torne pública a informação relacionada com a doença, acabam por consumir muito tempo e não são apropriadas para detetar infeções latentes uma vez que carecem de informação prévia [41]. Posto isto, Lim e outros autores deste estudo apresentaram uma abordagem *bottom-up* para a identificação deste tipo de infeções sem necessidade de informação prévia [41]. Eles propuseram um

modelo de *machine learning* não supervisionado capaz de identificar doenças infeciosas latentes através dos dados das redes sociais.

Existem alguns trabalhos com o propósito de prever o comportamento de uma pessoa utilizando apenas informação proveniente dos seus amigos nas redes sociais. Foi o caso de um estudo deu origem a um modelo probabilístico capaz de prever se uma pessoa irá adoecer e quando isso irá acontecer utilizando *tweets* e um classificador SVM. Os objetivos deste estudo foram alcançados com alta precisão e o modelo resultante é altamente escalável podendo ser aplicado à previsão de propriedades dinâmicas dos indivíduos em redes sociais vastas [45]. Os autores começaram por construir um classificador SVM para inferir o estado de saúde da pessoa através dos seus *tweets* e, depois, construíram o modelo *Conditional Random Field* (CRF) que pretendia prever o estado de saúde de um indivíduo a partir dos *tweets* e da localização de outras pessoas ligadas ao indivíduo, incluindo informação como o estado de saúde dessas pessoas e estimativas de encontros com elas [45]. Este estudo demarca a importância do fator localização na propagação de doenças infeciosas, do impacto da duração da colocação numa transmissão da doença e do tempo que medeia entre o contágio e o aparecimento dos sintomas [45]. Os autores consideram este modelo útil para desenvolver campanhas de prevenção, para orientar uma pessoa a monitorizar o seu estado de saúde tendo em conta as fragilidades do seu organismo e a exposição a determinados fatores que poderão desencadear a doença. Desta forma o indivíduo pode proteger-se evitando o contacto com esses fatores [45].

A utilidade da elaboração de sistemas capazes de detetar e prevenir surtos e doenças a partir de dados provenientes de redes sociais tem sido amplamente referida. Alguns estudos conseguiram rastrear com sucesso casos de doenças semelhantes à *influenza* através dos seus sintomas. Tais doenças apresentavam sintomas semelhantes aos da infeção por *influenza*, mas não eram suficientes para categorizar a doença como tal. O estudo conseguiu encontrar correspondência entre frases que usavam os termos *pior em casa, tosse noturna, dor de garganta e gripe suína,* e os surtos de doenças semelhantes à *influenza* que foram reportados no Reino Unido [47].

Um outro estudo mostrou que a aplicação das redes sociais, em especial do Twitter, à área da saúde vai mais longe estendendo-se inclusive à área das patologias comportamentais. De facto, García et al. desenvolveram uma metodologia para detetar e associar falsos perfis do Twitter que são usados para difamar um perfil de um utilizador real. Esta deteção é realizada

pela análise de conteúdo dos comentários em ambos os perfis e os autores conseguiram aplicar esta metodologia a um estudo de caso real que detetou e ajudou a travar um caso de *cyberbulling* numa escola [50].

Sadilek et al. também levaram a cabo um estudo focado nas relações e interações entre vários indivíduos e como estas afetavam a progressão do contágio de doenças. Utilizaram dados do Twitter e perceberam que para cada mensagem relacionada com saúde existem mais de 1000 mensagens não relacionadas com saúde. Os autores desenvolveram uma *framework* capaz de detetar eficazmente indivíduos doentes a partir do conteúdo das comunicações online [52]. A análise de *tweets* com *geo-tags* sugeriu que a ligação social a pessoas infetadas com sintomatologia aumentava significativamente a probabilidade de contrair a doença num futuro próximo [52].

Allen et al. [61] partiram de vários trabalhos já realizados onde se procura detetar surtos de gripe e outras doenças e associaram o método GIS para potencializar os efeitos dos modelos e ferramentas já existentes. Ao mesmo tempo, os autores apresentaram um processo de *machine learning* destinado a retirar o ruído dos *tweets* filtrando informação mais credível [61].

Um trabalho desenvolvido por Missier et al. tentou determinar o desempenho de duas abordagens usadas na deteção de surtos de dengue no Brasil a partir de dados do Twitter: classificação supervisionada e *clustering* não supervisionado. A primeira abordagem revelou um bom desempenho para várias classes: a classe *mosquito* foi considerada a mais legítima; a classe *doença* foi a mais informativa; a classe *noticias* consistia numa fonte indireta de informação; e a classe *piadas* correspondia a 20% dos *tweets* que foi considerada como ruído [66]. Na abordagem por clusters concluiu-se que havia menos controlo sobre o conteúdo do que o método tradicional de classificação. No entanto, como esta envolve muitas anotações manuais, acaba por ser mais custosa do que a abordagem por *clusters* [66].

Desafios

A utilização de DM com dados provenientes do Twitter para prevenção da doença na população tem sido amplamente estudada e utiliza como técnicas principais a classificação de *tweets*, a análise de sentimento e a classificação do utilizador. No entanto, estas técnicas apresentam algumas limitações. Por exemplo, a categorização de *tweets* em dados pode ser feita através de vários métodos, mas a maioria deles continua a depender da inteligência

humana para verificar os resultados. Não obstante, os métodos de comparação entre os resultados de classificação dos *tweets* carecerem de normalização [25]. O facto de a verificação depender largamente da inteligência humana também é propício a gerar erros.

Outra limitação é o tamanho e o propósito dos *tweets* que, como têm um número limitado de caracteres obrigam à utilização de métodos diversificados para conseguir extrair informação pertinente [39][62].

Também existem alguns estudos que sugerem que, embora os *tweets* relacionados com a gripe se correlacionem com a monitorização das doenças semelhantes à *influenza* proveniente do *Centers for Disease Control and Prevention* (CDC), nem sempre se confirma em laboratório que se trata efetivamente de casos de *influenza* [48].

A utilização de filtragem por palavras-chave para encontrar *tweets* relevantes é outro dos problemas das abordagens utilizadas pois ao recorrer a este método, os *tweets* que podem ser relevantes mas que não contenham as palavras-chave, são excluídos deixando de fora informação importante [37].

Já a utilização da geolocalização para filtragem de *tweets* pode gerar informação tendenciosa [38], devido à forma como a tecnologia é usada entre as áreas urbanas e rurais. O modo de vida e o quotidiano entre essas duas realidades tende a gerar informações e opiniões diferentes.

Assim, para que o Twitter possa ser considerado um fornecedor credível de dados para a monitorização da saúde, é necessário ultrapassar estas limitações [25]. Mas existem já alguns trabalhos que indicam que estas fontes de dados tem ganho credibilidade como por exemplo Zahem et al. que se dedicou especificamente a procurar publicações em redes sociais que pudessem ser consideradas confiáveis. Estes autores olharam não só ao conteúdo das publicações, mas ao próprio utilizador que as publicou e propuseram seis filtros que permitem atribuir um *rank* aos utilizadores de redes sociais e não apenas às suas publicações [49]. Desenvolveram assim uma aplicação de biomonitorização que extrai dados das redes sociais relacionados com bio-eventos e os processa de forma a encontrar os utilizadores mais fiáveis e as suas publicações que serão utilizadas noutras aplicações de biomonitorização [49].

Uma fonte de dados considerada credível e precisa reside na recolha de informação a partir de médicos, hospitais e laboratórios a partir dos relatórios gerados. No entanto, isto não permite detetar a emergência de um surto com a rapidez necessária [51].

Existe ainda o problema das ações maliciosas que podem ser levadas a cabo impactando negativamente o trabalho que se pretende fazer nesta área. Por exemplo, os *hackers* podem usar este tipo de dados para fins maliciosos como criar falsos alertas sobre surtos [46]. Se este comportamento, por si só, é negativo, a alternativa vem limitar a realização de trabalhos e investigações porque algumas entidades acabam por limitar o acesso às informações. Por exemplo, a Google não disponibiliza publicamente os termos usados para pesquisas relacionadas com a gripe para evitar que essa informação seja usada por *hackers* [46].

Para ultrapassar estes obstáculos, alguns autores consideraram que são necessários novos e melhores algoritmos de DM e também máquinas com grande poder computacional para anular o ruído gerado pela grande quantidade de dados [51]. Os métodos de análise de séries temporais podem ser úteis para analisar vários anos de dados e comparar o nível da epidemia é maior ou menos que o ano anterior. A existência de plataformas web e aplicações móveis onde a população possa relatar eventos para as entidades de saúde também poderá ser uma mais-valia neste campo [51].

Mas existem também alguns trabalhos que procuram confirmar a precisão/exatidão dos dados provenientes de redes sociais aplicados à área da saúde. Por exemplo, um estudo realizado pelo ICF Macro percebeu que as publicações referentes a duas questões de saúde ambiental (perclorato na comida de bebé e problemas com fungos no revestimento de paredes usado em algumas casas da China) no Facebook e em *blogs* tinham uma boa correspondência com o que constava nos relatórios oficiais sobre os mesmos assuntos [46].

Já autores como Al-Garadi et al. consideraram que poderá nunca ser possível ter um sistema de monitorização baseado em dados de redes sociais suficientemente eficaz para substituir os meios tradicionais e que o máximo que se conseguirá atingir são meios complementares de fornecimento de dados que devem ser integrados com os dados recolhidos pelos métodos tradicionais [56].

2.6. Conclusão

Os dados provenientes de redes sociais contêm informação muito valiosa para a áreas da saúde incluindo a possibilidade de auxiliar o rastreio de epidemias mais rapidamente do que os métodos tradicionais (como a elaboração de relatórios oficiais pelos profissionais de saúde, por exemplo) [56]. Outra das vantagens de utilizar este tipo de dados tem a ver com o facto de ser possível adicionar informação geográfica e temporal para a análise e ter uma perceção mais precisa da propagação das doenças [56]. Muitos dos estudos encontrados na literatura visam a elaboração de modelos para detetar e tentar prevenir surtos de *influenza* e gripe suína. Estes modelos podem, no entanto, ser reformulados para aplicar à deteção e previsão de surtos relacionados com outras doenças [53][56].

Como vimos na secção *2.5.1*, existem ainda algumas limitações a ultrapassar para que possamos chegar a resultados mais fidedignos. No entanto, na literatura parece existir um consenso relativamente à importância da utilização do Twitter na área da saúde, em particular, aplicado ao caso da deteção de surtos pois este fornece uma grande quantidade de dados em tempo real.

Embora existam algumas falhas e limitações, no geral, as redes sociais parecem ser uma boa alternativa para melhorar a saúde pública e identificar as populações alvo de intervenção.

3. Tecnologias de *Machine Learning*

3.1. Introdução

A quantidade de dados que circunda as nossas vidas tem vindo a crescer exponencialmente o que se deve, em grande parte, aos computadores e aos avanços que se têm feito nesta área, por exemplo, ao nível da memória de armazenamento e da Internet [70]. De facto, os discos externos encontram-se cada vez mais baratos e a emergência do armazenamento em *cloud* permite-nos guardar cada vez mais dados e até a computação ubíqua tem assumido o seu papel em guardar as nossas decisões, hábitos financeiros, escolhas no supermercado, viagens, etc. [70]. Além disso, a Internet possibilita que uma grande quantidade de informação esteja acessível a qualquer pessoa e possa ser usada para diversos fins, incluindo, em estudos de DM.

Como veremos neste capítulo, a análise de grandes volumes de dados, em especial de forma exploratória, constitui um método poderoso transversal a diversas áreas da nossa vida, incluindo na área da saúde onde os avanços tecnológicos têm impulsionado cada vez mais novas formas de diagnóstico, do combate a doenças e da sua prevenção.

O *Machine Learning* (ML) possui inúmeros métodos capazes de auxiliar esta tarefa e já existem várias *frameworks* que providenciam formas mais intuitivas, escaláveis e otimizadas de processar grandes quantidades de informação, permitindo tirar ilações e construir informação pertinente em pouco tempo quando comparadas com os métodos tradicionais de análise de dados. Os dados poderão ser provenientes das mais diversas fontes (como por exemplo satélites, televisão, rádio, transações diversificadas, etc.) de onde se destacam os dados provenientes da *web* que consistem também o foco deste trabalho. A extração de dados a partir da *web* providencia uma grande quantidade de dados passíveis de serem analisados e transformados em informação e que podem incidir em diversas áreas das nossas vidas. Por exemplo, atualmente é muito usual as empresas examinarem *hiperlinks* das páginas web para medir a importância dos websites para as pessoas [70]. A análise de *queries* dos motores de pesquisa (já abordada no tópico *2.2.2*) constitui outra dessas fontes.

Como foi referido anteriormente, as redes sociais promovem a partilha de informação de forma massiva. Nelas as pessoas partilham muita informação pessoal sobre os seus sentimentos, opiniões, ideologias, etc. e esta partilha pode assumir a forma de comentários de texto, vídeos, gostos musicais ou cinematográficos, locais visitados entre outros [70]. Mas

como é que estas fontes e estes dados são trabalhados de forma a extrair deles informação relevante para determinados propósitos? É esta pergunta que procuraremos responder neste capítulo.

O presente capítulo encontra-se estruturado da seguinte forma: *3.2*, onde é apresentado o conceito de ML e como ele se insere no processo de DM bem como uma breve descrição acerca deste processo; *3.3*, que versa sobre os algoritmos e modelos mais predominantes nesta área distinguindo entre os algoritmos de aprendizagem supervisionada e os algoritmos de aprendizagem não supervisionada; *3.4*, um tipo particular de algoritmos de aprendizagem, que fornece uma breve descrição sobre a estrutura de uma rede neuronal e apresenta diversos exemplos de implementação destas redes; *3.5*, onde é efetuada uma breve referência e descrição de algumas das *frameworks* utilizadas para facilitar e otimizar os processos de ML; *3.6*, que expõe uma breve apresentação sobre o tipo de aplicações que os métodos de ML permitem nas áreas da saúde dando alguns exemplos do que já existe e do que se pretende alcançar; e *3.7*, que apresenta uma breve reflexão sobre os tópicos abordados neste capítulo.

3.2. Machine Learning

Definição

Como foi referido no Capítulo 2, o DM é o processo de extrair informação útil que se encontra implícita ou desconhecida a partir de dados. Assim, constroem-se modelos que procuram a existência de padrões entre os dados que possam ser usados para fazer previsões sobre dados futuros. A Figura 5 ilustra o processo de DM.

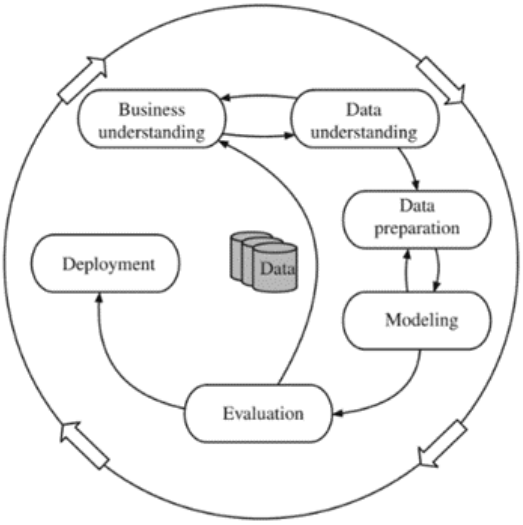

Figura 5 - Ilustração das fases do processo de DM [70].

É impossível expor no presente relatório todas as componentes e complexidade que envolvem o processo de DM. No entanto, podemos fazer uma breve referência às fases que o constituem de forma a clarificar este conceito. Antes de mais, é importante perceber que o processo de DM é cíclico, isto é, que todas as fases se encontram interligadas e que no final de cada fase poderemos voltar ao início do processo [93]. Tal acontece porque só à medida que se avança no processo é que vamos compreendendo melhor os dados, vão surgindo questões a esclarecer e vários aspetos que não foram contemplados no início do processo. O ciclo de vida apresentado na Figura 5 foi definido por um modelo de referência denominado Cross Industry Standard Process for Data Minning (CRISP-DM) [70].

A *compreensão do negócio* (*business understanding*) pretende responder à pergunta "*Qual é o objetivo a alcançar?"*. Ou seja, é a fase em que se define para que se pretende utilizar o DM. Para tal, é necessário: compreender os objetivos e requisitos do negócio, se estes podem ser alcançados através deste processo e estabelecer que tipo de dados recolher para construir um modelo [93].

Na fase de *compreensão dos dados (data understanding)* constrói-se um *dataset* inicial que vai ser estudado para perceber se é útil ou não continuar com o processo. Muitas vezes os dados disponíveis são insuficientes ou incompletos e é necessário encontrar fontes de dados alternativas para conseguir avançar com o processo. É por isto que nesta fase poderá ser necessário voltar à fase anterior e reajustar ou redefinir o objetivo [70].

Os dados compreendidos na fase anterior encontram-se em estado bruto e têm que ser processados para que os algoritmos consigam produzir um modelo, o que acontece na fase de *preparação dos dados*. No entanto, o que acontece frequentemente é que a *preparação dos dados* pede ajuda à *modelação,* pois a fase de processamento pode necessitar de construir modelos a partir dos dados para depois os transformar. Assim, estas duas fases estão interligadas e os resultados obtidos na *modelação* impactam grandemente, e obrigam a redefinir, nas técnicas de processamento escolhidas na fase anterior [70].

A fase de avaliação é aquela que garante a qualidade dos modelos obtidos. Nela, procura-se perceber se os modelos obtidos permitem fazer previsões ou não e existem diversas técnicas que podem ser usadas para alcançar este objetivo tendo sido algumas delas descritas na secção *2.2.2* e outra que serão referidas no tópico *3.3*.

Na verdade, as fases de *preparação dos dados*, *modelação* e *avaliação* constituem o núcleo dos métodos de ML, ou seja, as técnicas que sustentam o processo de DM [70]. Este é utilizado para extrair informação a partir de dados em bruto. Esta informação será compreensível e poderá ser aplicada a vários propósitos. O ML envolve um processo de abstração que implica inferir qualquer estrutura subjacente aos dados, isto é, descobrir padrões estruturais nos dados [70]. Um padrão estrutural poderá fornecer um conjunto de regras ou umas árvores de decisão que serão usadas para ajudar o ser humano a tomar uma decisão. Para encontrar um padrão estrutural é necessário um conjunto de *inputs* sobre os quais se vão gerar todas as combinações. Um *input* pode ser um conceito, uma instância ou um atributo. O que se pretende aprender será a descrição do conceito. O material fornecido para aprendizagem materializa-se em instâncias (cada uma delas com o seu próprio conceito) e os atributos são o que caracteriza cada instância (são medidas de várias características da instância) [71]. Os *outputs* constituem resultado da aprendizagem e podem assumir variadíssimas formas.

O ML encontra-se bastante ligado à área da estatística. Ambos os campos utilizam técnicas de análise de dados. Na verdade, podemos considerar que existe uma linha contínua de técnicas de análise de dados sendo que umas derivaram dos cursos de estatística e outras estão mais associadas ao tipo de ML possibilitado pelas ciências computacionais [71]. No entanto, podemos considerar que a estatística é uma disciplina mais centrada em testar hipóteses ao passo que o ML procura formular o processo de generalização procurando a

partir das hipóteses possíveis [70]. As duas áreas acabam por se complementar e a estatística é usada para validar os modelos obtidos a partir do ML e os seus algoritmos.

O ML é, portanto, um conjunto de técnicas de aprendizagem a serem utilizadas no processo de DM [93]. Na secção seguinte apresentamos alguns desses métodos e modelos.

3.3. Tipos de algoritmos em *Machine Learning*

Existem várias formas de categorizar os algoritmos utilizados em ML mas podemos inseri-los nas seguintes categorias: aprendizagem supervisionada; aprendizagem não-supervisionada; aprendizagem semi-supervisionada e aprendizagem por reforço.

Na aprendizagem supervisionada, o *output* esperado ao fornecer um determinado *input* já é conhecido, ou seja, já sabemos o resultado que pretendemos alcançar dado um determinado *input*. Assim, este tipo de aprendizagem pressupõe que existe uma relação entre os dados de entrada e os dados de saída e que se *A* ocorrer, então deverá ocorrer *B*, sendo *B* conhecido *a priori* [71]. No fundo, a aprendizagem supervisionada visa prever uma variável dependente a partir de uma variável independente.

A aprendizagem não supervisionada pode ser aplicada a procuras exploratórias pois permite responder a problemas em que não sabemos qual será o *output*. Assim, perante um conjunto de dados, podemos extrair padrões estruturais ou modelos que desconhecemos à partida [71].

As redes neuronais são um exemplo e modelos que, de facto, se inserem nestas duas categorias o que vai depender do output desejado [77]. Como os algoritmos utilizados neste projeto provêm de redes neuronais estas merecem uma explicação mais aprofundada que pode ser consultada na secção *3.4*.

A aprendizagem semi-supervisionada é um meio-termo entre as duas anteriores combinando-as para alcançar os resultados desejados. É útil para problemas do mundo real onde os resultados dos inputs podem ser conhecidos ou não, ou seja, os dados podem ou não ter rótulos [71].

A aprendizagem por reforço visa descobrir a melhor resposta ou ação a executar perante as circunstâncias [94]. Não existe conhecimento *a priori* sobre o tipo de resposta, pelo que o agente é exposto a um determinado ambiente e aprende por tentativa e erro. Através da experiência passada, o agente procura escolher a melhor ação a executar perante determinadas circunstâncias. A Figura 6 ilustra de forma simples este fluxo.

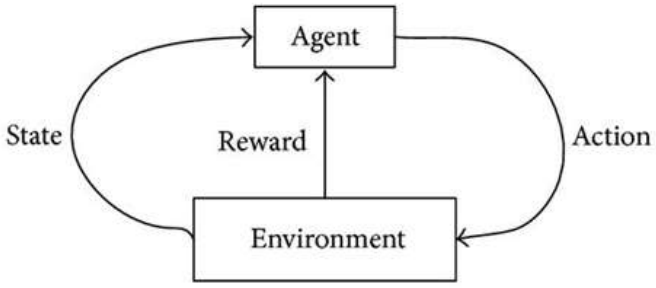

Figura 6 - Ilustração do fluxo de aprendizagem por reforço [76].

Dado o propósito e o enquadramento do presente trabalho, iremos focar-nos nas duas primeiras abordagens que descrevemos já de seguida.

3.3.1. Algoritmos de aprendizagem supervisionada

Nesta secção serão apresentados de forma breve alguns algoritmos de aprendizagem supervisionada mais comummente usados: regressão linear, aprendizagem baseada em instâncias, aprendizagem baseada em múltiplas instâncias, classificador de *Bayes* e *SVMs*.

Regressão linear

A regressão linear é um método emprestado da estatística que procura relações entre duas ou mais variáveis partindo de dados existentes para construir um modelo que permita prever dados futuros. A regressão linear pode ser classificada em: *Regressão Linear Simples* (RLS) ou *Regressão Linear Múltipla* (RLM). Enquanto a RLS encontra relações de um para um, a RLM encontra relações de um para muitos [70]. Os algoritmos de regressão linear procuram, então, identificar correlações entre variáveis podendo essa correlação ser positiva ou negativa. A regressão procura descobrir a função que expressa a existência de correlações ou padrões nos dados.

Um caso particular de regressão linear é o de Regressão Logística que foi utilizado neste projeto para construir modelos. A diferença, é que na Regressão Logística a variável dependente é binária.

Aprendizagem baseada em instâncias

Este tipo de aprendizagem difere dos demais porque, em vez de originar um modelo descritivo, ele armazena os exemplos de treino e parte para a generalização quando surge a

classificação de uma nova instância. Para estes métodos, as instâncias são pontos num espaço euclidiano [71]. Um exemplo da representação de um determinado tipo de flor no espaço euclidiano pode ser observado na Figura 7.

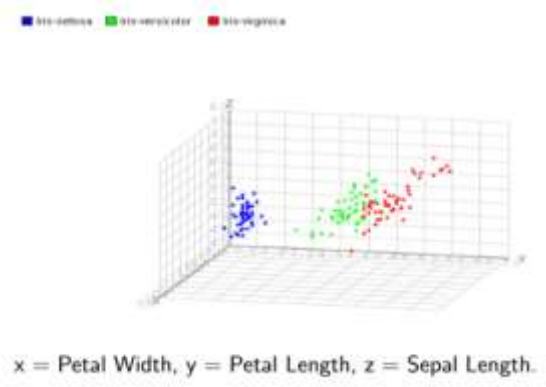

Figura 7 - Espaço euclidiano representativo das dimensões de um tipo de flor (largura da pétala, comprimento da pétala e comprimento da sépala [73].

A aprendizagem por instâncias vai apenas guardar os exemplos de treino e depois procura um conjunto de instâncias semelhantes para classificar uma nova instância. O resultado da aprendizagem é um conjunto de distâncias entre a nova instância e os exemplos de treino. Existem vários algoritmos para executar esta tarefa, mas o mais elementar é o algoritmo *k-NN* onde as instâncias são representadas por pontos num espaço n-dimensional e utiliza a regra dos *vizinhos mais próximos* para classificar a nova instância que é considerada a mais frequente entre as *k* amostras mais próximas [71]. Uma das vantagens deste método é que toda a informação dos exemplos de treino fica guardada, mas tem a desvantagem de poder tornar-se computacionalmente pesado pois todo o processamento é feito durante a classificação.

Aprendizagem baseada em múltiplas instâncias

A aprendizagem baseada em múltiplas instâncias é um método de aprendizagem supervisionada utilizado quando existe incompletude nos rótulos (*labels*) dos exemplos de treino. Neste tipo de aprendizagem, em vez de ser fornecido um conjunto de instâncias individualmente rotuladas, o algoritmo recebe um conjunto de *bags* (conjuntos de instâncias) com um rótulo. Cada *bag* contém várias instâncias. Existem várias formas de implementar este tipo de aprendizagem sendo a mais simples a classificação binária. Neste tipo de classificação, a categorização do *bag* será positiva se contiver pelo menos uma instância positiva, ou negativa se não contiver nenhuma [74]. A Figura 8 mostra um exemplo de como poderá ser feita a categorização atrás descrita.

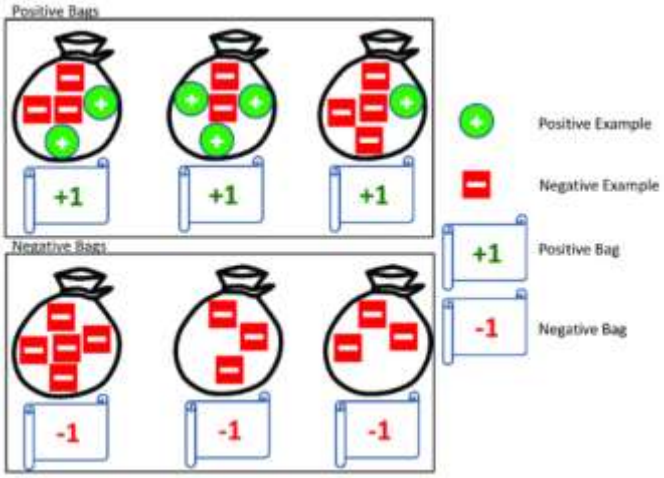

Figura 8 - Ilustração do conceito de aprendizagem baseada em múltiplas instâncias [75].

A partir desses *bags* o algoritmo pode realizar uma destas tarefas: induzir o conceito para rotular cada instância ou aprender como rotular as instâncias sem aprender o conceito [74].

Classificador de *Bayes*

O classificador de *Bayes*, quando aplicado a um conjunto de treino (conjunto de dados) e a um conhecimento *a priori* dá a classe/hipótese mais provável, ou seja, dá-nos a classificação ótima de um padrão de dados de acordo com um determinado critério [74]. Portanto, permite calcular *a posteriori* a hipótese máxima *a priori* (MAP).

Support Vector Machine (SVM)

Este é um dos métodos utilizados para classificação no presente projeto onde foi escolhida a sua versão *LinearSVC*. Este método utiliza a análise de regressão e a classificação procurando reconhecer padrões dado um conjunto de dados. Perante um conjunto de dados, é capaz de predizer a qual uma de duas classes esse conjunto pertence sendo um classificador binário. Os exemplos de treino são representados como pontos num espaço onde os exemplos de cada categoria estão divididos por um espaço evidente. Assim, ao receber um novo conjunto de dados, o SVM coloca-os no mesmo espaço e prediz a categoria em que eles se inserem [70]. A Figura 9 mostra uma representação de um espaço com duas classes distintas.

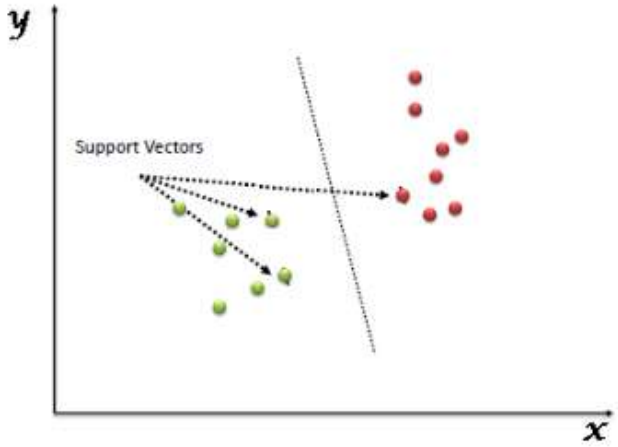

Figura 9 - Representação de um espaço com duas classes utilizando o SVM [78].

O SVM é um excelente método para domínios onde existe uma margem de separação evidente. No entanto, não é um bom método a utilizar quando os dados têm muito ruido o que é um problema se houver sobreposição de classes, quando o conjunto é muito grande (pois aumenta muito a complexidade computacional) [71].

3.3.2. Algoritmos de Aprendizagem não supervisionada

Alguns dos algoritmos mais utilizados em aprendizagem não supervisionada são: regras de associação, os algoritmos de *clustering* e o *dividir para conquistar*. São estes que apresentamos de seguida.

Regras de associação

As regras de associação são padrões descritivos que indicam se a existência de um conjunto de itens numa base de dados está associada à presença de um outro conjunto na mesma base de dados [70]. Esta técnica de análise também é chamada de *Market Basket Analysis* porque teve a sua origem na análise de dados relativos a cestos de compras num supermercado para descobrir conjuntos de produtos que costumavam ser adquiridos frequentemente. Numa regra de associação há dois conjuntos de itens: um antecedente (LHS) e um consequente (RHS) onde se o antecedente existir, então existe o consequente chama-se comprimento do conjunto ao número de itens em cada conjunto [95]. Para discriminar as associações mais relevantes recorre-se a várias medidas, entre as quais se destacam: o suporte, ou a contagem das transações onde todos os itens aparecem nos dois conjuntos (antecedente e consequente); a confiança, que indica a probabilidade condicional de ocorrer B quando A ocorreu; os itens

frequentes, que são os itens que se repetem com frequência elevada; e os itens raros, que constituem os conjuntos de itens que se manifestam numa percentagem reduzida [95].

Clustering

O objetivo dos algoritmos de *clustering* é agrupar os dados fornecidos de acordo com o grau de semelhança de um determinado critério formando *clusters* (grupos de dados). Dada a sua natureza, o *clustering* é muito utilizado para encontrar padrões inesperados entre os dados. É um método muito utilizado em extrações de texto onde o agrupamento dos dados visa juntar textos que versam sobre um mesmo tema [70].

Existem muitos algoritmos de *clustering* que podem categorizar-se em vários tipos: os métodos de partições, os baseados em densidade, os baseados na teoria dos grafos, os hierárquicos, os baseados em *grids*, os baseados em técnicas de procura combinatória, os baseados em redes neuronais, entre outros [71][96][70].

Os métodos de partições, por exemplo, visam construir partições/grupos de objetos e avaliar cada partição de acordo com um critério. Um exemplo deste tipo de método é o algoritmo *K-Means* onde perante um conjunto de dados com *n* instâncias constrói *k* partições/grupos (onde *k* é definido *a priori*) de dados sendo sempre $k<=n$ [70]. Já o *clustering* hierárquico utiliza um critério escolhido para decompor e organizar os objetos dando origem a uma árvore onde os objetos são representados por folhas e os pontos, nós das folhas, contêm a estrutura de semelhança entre os pontos [96].

O *clustering* tem a vantagem ser um ótimo método de exploração de dados, mas os seus resultados dependem francamente da correta representação dos dados e das métricas de semelhança eleitas.

Dividir para conquistar e árvores de decisão

Este método consiste em, recursivamente, dividir um problema em problemas mais pequenos que voltam a ser divididos até atingir o mínimo tamanho possível. Os resultados destes problemas são combinados até encontrar a solução para o problema maior [70]. A pesquisa binária (algoritmo de pesquisa), o *quicksort* (algoritmo de ordenação) e os *pares mais próximos*, por exemplo, são alguns algoritmos deste tipo. A Figura 10 ilustra o processo do método *dividir para conquistar* que se encontra dividido em três passos: dividir (partir o problema em problemas menores recursivamente), conquistar (receber muitos sub-

problemas cada um representando uma solução) e combinar (ao resolver o problema mais pequeno, combinam-se os vários problemas par chegar à solução).

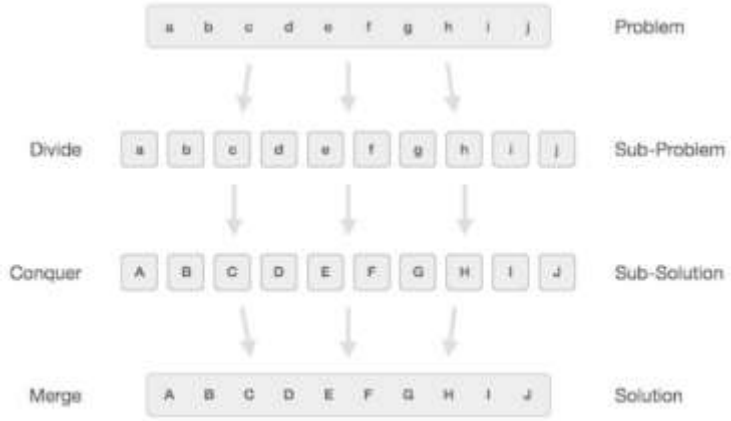

Figura 10 - Ilustração do processo do método *dividir para conquistar* [79].

3.4. Redes neuronais

As redes neuronais, também utilizadas neste projeto, são modelos inspirados no Sistema Nervoso Central (SNC) compostas por vários neurónios artificiais que se encontram interligados formando um sistema. Tal como acontece no SNC, também nas redes neuronais, os neurónios recebem/percecionam entradas do ambiente, processam-na e produzem uma resposta [97]. São muito úteis para reconhecer padrões e conseguem resolver problemas que os métodos baseados em regras não podem resolver. Como são análogas ao SNC, estes modelos são capazes de perceber o meio ambiente e aprender melhorando o seu desempenho utilizando um processo de ajustamento de pesos durante o treino [97]. Considera-se que a rede aprendeu quando é capaz de generalizar uma solução para uma classe de problemas. Tal como foi referido anteriormente, as redes neuronais podem inserir-se no âmbito da aprendizagem supervisionada, no âmbito da aprendizagem não supervisionada, ou no âmbito da aprendizagem por reforço dependendo da forma como se relacionam com o ambiente [70]. Relembrando estes conceitos, agora aplicados ao caso particular das redes neuronais, temos as seguintes aplicações gerais:

- Aprendizagem supervisionada – perante um padrão de entrada há um agente externo que comunica à rede neuronal a resposta correta para esse padrão. É o que acontece nas tarefas de classificação, onde a rede recebe conjuntos de dados com rótulos e irá aprender correlações entre os rótulos e os dados. Alguns exemplos concretos deste tipo de aplicação são:

  - Reconhecimento de imagem;
  - Reconhecimento facial e de expressões faciais;
  - Reconhecimento de gestos;
  - Deteção e reconhecimento de voz;
  - Tradução de voz para texto;
  - Classificadores de spam ou fraudes a partir do reconhecimento de texto;
- Aprendizagem não-supervisionada – a rede neuronal terá que descobrir a resposta para os dados de entrada na ausência de um agente externo que lhe confirme essa resposta. Um exemplo já mencionado anteriormente é o *clustering* que permite agrupar dados tendo em conta semelhanças entre eles. É uma abordagem muito consensual com o mundo real onde muitos dados permanecem desconhecidos pelo que quanto maior a quantidade de dados que o algoritmo recebe, mais precisa será a aprendizagem. É muito usado em procura de semelhanças (em textos, imagens, etc.) e na deteção de anomalias, uma vez que ao identificar semelhanças, conseguem identificar diferenças e comportamentos inesperados.
- Aprendizagem por reforço – a rede neuronal aprende por tentativa em erro e existe um agente externo que avalia a sua resposta.

As redes neuronais são compostas por camadas sendo cada camada composta por nós ou neurónios. Cada nó assemelha-se a um neurónio biológico na medida em que é ativado perante um dado estímulo, ou seja, recebe entradas de dados com um conjunto de pesos. Estes pesos tornam a entrada num estímulo de intensidade maior ou menor para o neurónio. Após a soma dos pesos, o resultado é passado para uma função de ativação que decide se esse estímulo deve progredir pela restante rede ou não e produzir uma classificação. A Figura 11 mostra um exemplo de um neurónio biológico e de exemplo de um neurónio artificial simples, também conhecido como *percetrão*.

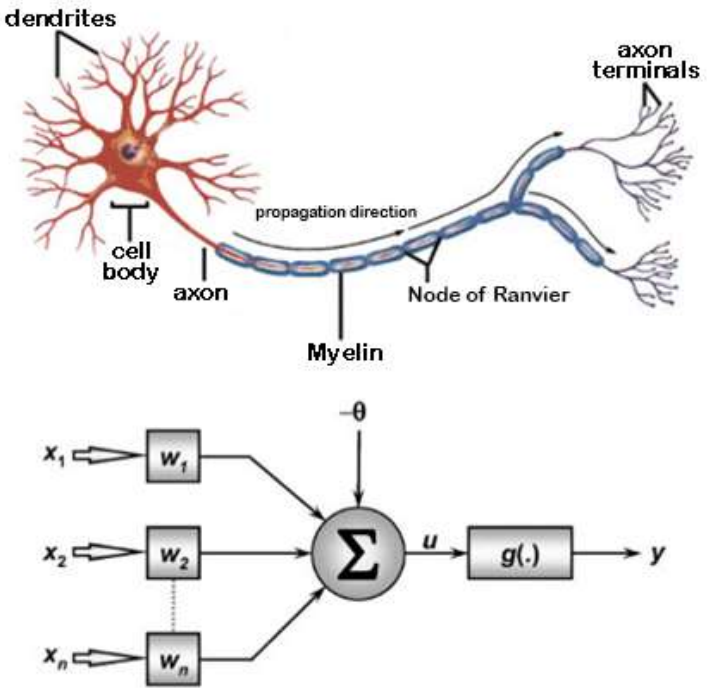

Figura 11 - Ilustração de neurónio biológico e neurónio artificial [81].

Na Figura 11, os valores de x_n, correspondem aos dados de entrada que são multiplicados pelos pesos w_n, passando depois por uma função soma (∑) que adiciona todos os produtos dando o potencial de ativação (u) utilizando o *bias* (valor constante) para aumentar ou diminuir a entrada de u. A Figura 12 é uma representação simples de uma rede de nós com três camadas: a camada de entrada (*input layer*), a camada escondida (*hidden layer*) e a camada de saída (*output layer*).

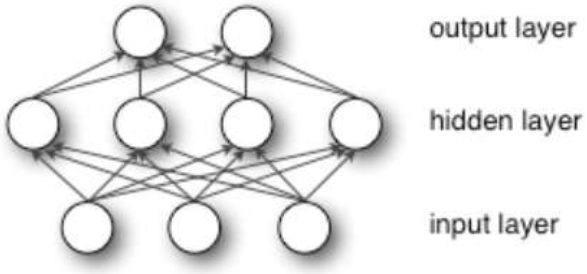

Figura 12 - Rede de nós com três camadas [80].

As redes neuronais são utilizadas em *deep learning* permitindo simular o comportamento do SNC formando grandes redes artificias [80]. O *deep learning* consiste, na realidade num

processamento de dados muito superior àquele que é conseguido com uma rede neuronal comum recorrendo a serviços de armazenamento em nuvem e em Unidades de Processamento Gráfico (GPU's) [97]. As redes neuronais profundas têm mais do que uma camada escondida, portanto, são redes neuronais com mais do que 3 camadas. Nestas redes, cada camada de nós recebe da camada anterior um conjunto de características diferentes e treina esse conjunto que se torna mais complexo quanto mais avança na rede. Isto é conseguido através da combinação e agregação de características provenientes da camada anterior. A este fenómeno dá-se o nome de *hierarquia de características*, hierarquia esta que se torna mais complexa e exige maior nível de abstração quanto mais se avança na rede [82]. Assim, é possível treinar com *datasets* de muitas dimensões e encontrar estruturas até então desconhecidas dentre dados não estruturados e não rotulados, como é o caso de dados provenientes de fontes media (imagens, áudio, vídeo, etc.) e da maioria dos problemas no nosso mundo [71]. É também o que acontece com dados provenientes de texto onde as redes neuronais profundas conseguem separar textos de acordo com o seu teor ou tema em diferentes categorias (por exemplo, separar emails considerados *spam*, dos outros).

É muito importante que as redes neuronais tenham, tal como os humanos, capacidade de generalização, isto é, de aplicar a outros contextos aquilo que aprenderam com a sua experiência. Desta forma, as redes neuronais artificiais devem ser capazes de ter sucesso com dados desconhecidos. Muitas vezes, o que ocorre com redes mais complexas é que são treinadas em demasia levando a um ajuste de pesos desadequado, problema este que é conhecido como *over-training*. Por outro lado, as redes mais simples podem não ser capazes de modelar um problema complexo [97].

Em suma as redes neuronais têm muitas vantagens no que concerne à exploração, aprendizagem de padrões desconhecidos e reconhecimento e podem ser aplicadas em inúmeras áreas para resolver os mais variados problemas da vida real. Por exemplo, utilizando dados históricos, elas conseguem aprender e identificar comportamentos normais ou desejáveis separando-os dos comportamentos anómalos e esta informação pode ser usada na área da saúde. Uma enorme vantagem é o facto de estas redes conseguirem tudo isto de forma automatizada sem necessidade de intervenção humana.

De seguida apresentamos alguns exemplos de redes neuronais frequentemente utilizados.

3.4.1. Redes neuronais *feedforward*

É o tipo de rede onde cada camada está ligada à camada seguinte, mas a comunicação segue apenas uma direção que começa na camada de entrada e termina na camada se saída. Podemos ver um exemplo de uma rede *feedforward* com duas camadas escondidas na Figura 13.

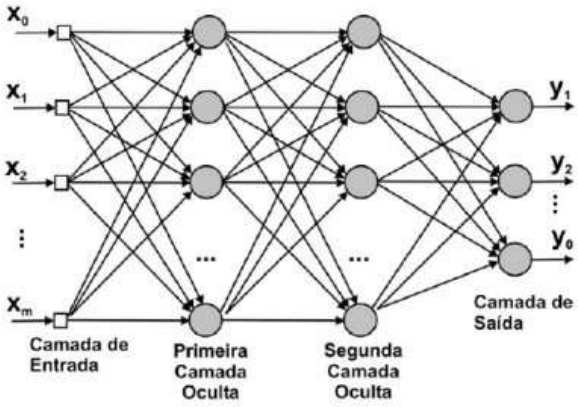

Figura 13 - Exemplo de rede *feedforward* [82].

Neste tipo de rede, as saídas de uma camada constituem as entradas da camada seguinte. Para estruturar este tipo de rede é necessário determinar: a camada de entrada, as camadas escondidas e a camada de saída. Esta definição carece de alguns testes para escolher a estrutura ótima da rede [97].

3.4.2. *Backpropagation*

O *backpropagation* consiste, na verdade, no modo de treino da rede sendo um modo de aprendizagem supervisionado que consiste em calcular o erro e acertar os pesos em todas as camadas. Este procedimento é efetuado na direção oposta à da rede *feedforward*, ou seja, parte da camada de saída até chegar à camada de entrada [98].

3.4.3. *Recurrent Neural Networks* (RNN)

As RNN são o tipo de rede neuronal que permite ver um comportamento dinâmico ao longo do tempo. Nela, as ligações entre os neurónios materializam um gráfico numa sequência temporal e conseguem usar a sua memória interna para processar entradas de forma sequencial [97]. Esta capacidade de memória está relacionada com o facto destas redes permitirem ciclos porque, na verdade, é como se uma destas redes fosse composta por várias cópias de si mesma onde os dados passam de uma cópia para a seguinte, de forma sequencial

pelo que funcionam bem com dados provenientes de listas [83]. A Figura 14 ilustra bem este conceito.

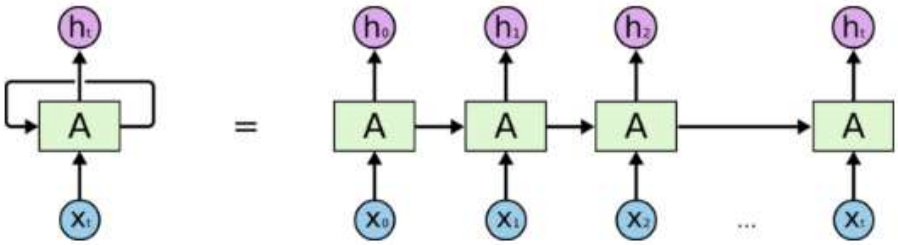

Figura 14 - Exemplo do comportamento de uma RNN [83].

Neste tipo de rede neuronal, tanto o seu *input* como o *output* são vetores. É possível utilizar o *backpropagation* com estas redes para saber em que direção ajustar os pesos, como veremos no tópico *3.4.1*. O tipo de aplicação desta rede pode ir desde o reconhecimento de escrita passando pelo reconhecimento de discurso, tarefas de tradução, etc.

3.4.4. *Long Short-Term Memory (LSTM)*

As LSTM são redes que consistem numa variação das RNN que mantém o erro permitindo que este seja *retropropagado* (*backpropagated*) através das camadas possibilitando associar causas e efeitos remotamente [99]. Estas redes contém um tipo de memória pois a informação pode ser guardada e acedida (escrita e lida) a partir de um neurónio e são estes quem resolve se a sua informação pode ser acedida. É como se cada neurónio tivesse *portas* que abrem e fecham consoante os sinais que recebem deixando passar ou não a informação [99]. Essa informação é filtrada de acordo com os seus pesos que são também ajustáveis.

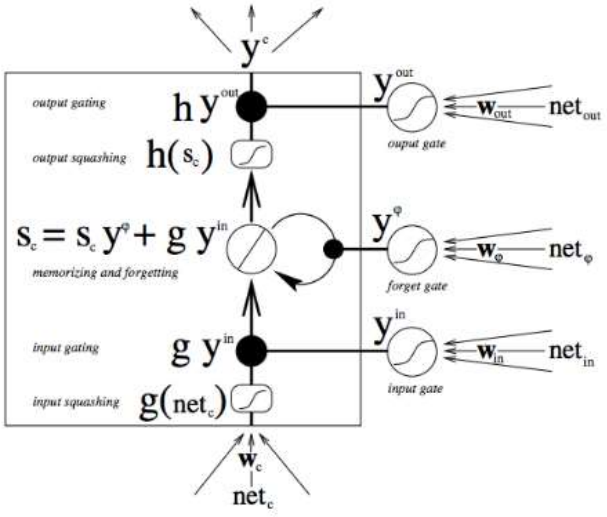

Figura 15 - Exemplo do fluxo de dados num neurónio com memória [80].

A Figura 15 demonstra como a informação é controlada através das *portas* do neurónio. Existem quatro *portas* neste sistema: a *porta forget*, que decide que informação é descartada e que informação é guardada (o que se deve manter dos estados anteriores); a *porta* de *input*, que faz o *update* à informação (a informação de deve ser adicionada ao estado corrente); a porta do neurónio (*cell gate*), onde é calculado o estado do neurónio; e a *porta* de *output*, que decide qual será o próximo estado escondido [80].

3.4.5. *Convolutional Neural Network* (CNN)

A CNN é uma rede do tipo *feedforward* onde são usadas camadas de neurónios múltiplas. Ela recebe como *input* uma imagem e atribui importância a várias características da imagem de forma a distingui-las [97]. A CNN consegue captar as dependências temporais e espaciais de uma imagem e reutiliza os pesos reduzindo as imagens a uma forma de fácil *processamento.*

3.4.6. *Gated Recurrent Unit* (GRU)

A GRU é uma rede semelhante à LSTM baseando-se no mesmo conceito de *portas* mas tem apenas duas: uma *porta* de *reset* e uma porta de *update*. A primeira é uma porta onde se decide quanta da informação será esquecida e a segunda decide que informação se vai descartar e que informação se vai reter. É um tipo de rede mais rápida do que a LSTM e não tem os estados escondidos como esta [100].

3.4.7. *Random Forest*

Esta é uma das redes neuronais utilizadas para treinar modelos no presente projeto. Trata-se de um tipo de algoritmo utilizado para classificação e regressão. Constrói diversas árvores de decisão durante o treino [70]. O resultado é a classe que representa a moda das classes, no caso da classificação (que é a tarefa utilizada no projeto). Este tipo de algoritmo tem a vantagem, de reduzir o *overfiting* característico das árvores de decisão.

3.4.8. *Multilayer Perceptron*

Outra rede neuronal escolhida para treinar modelos no Sistema de deteção de surtos foi a *Multilayer Perceptron*. À semelhança do *Randoom Forest*, esta rede neuronal também pode ser usada para classificação e para regressão, sendo que no presente projeto se utilizou para classificação. Trata-se de uma rede semelhante a um perceptrão mas com mais do que uma camada escondida. A aprendizagem nesta rede é feita através do *Backpropagation* mencionado anteriormente.

3.5. *Frameworks* de *Deep Learning*

Como pudemos perceber pelos subcapítulos anteriores, os algoritmos de *deep learning* estão cada vez mais complexos e representam uma mais-valia importante nos processos de exploração de dados. Enquanto continua a ser possível desenvolver os próprios algoritmos de raiz, tal consome muito tempo e não é uma solução prática para modelos como o RNN, por exemplo. Assim, as *frameworks* de *deep learning* ajudam na tarefa de implementar redes neuronais. Nesta secção, apresentamos algumas delas: a *TensorFlow*, a *Keras*, a *Apache Mahout*, a *Scikit-learn* e a *Apache Spark* (utilizada no presente projeto).

3.5.1. *TensorFlow*

A TensorFlow[15] é uma *framework* de *deep learning open source* desenvolvida pela Google que consiste numa biblioteca de algoritmos de *deep learning*. Utiliza grafos de *dataflow* para construir modelos [101]. É um sistema que opera a larga escala em ambientes heterogéneos [84]. A *TensorFlow* mapeia os nós presentes nesses grafos em várias máquinas de *clustering* e, dentro de cada uma delas, em vários dispositivos computacionais que no seu conjunto compõem as Tensor Processing Unit (TPUs). Trata-se de uma arquitetura composta por Computer Process Units (CPUs) multi core, os GPU e os Application Specific Integrated

[15] https://www.tensorflow.org/

Circuits (ASICs) [84]. Graças a esta arquitetura, a *TensorFlow* é muito flexível e permite algoritmos de treino muito otimizados. O grafo de *dataflow* é usado para representar a computação num algoritmo e o estado em que cada algoritmo opera [84]. De acordo com Abadi et al. este sistema foi inspirado nos modelos de sistemas de *dataflow* utilizados em programação de alto nível e em ferramentas de *parameter server* de baixo nível de eficiência [84].

A *TensorFlow* distingue-se dos demais sistemas de *dataflow* porque os seus vértices representam computações que têm um estado próprio que pode ser alterado. As arestas estabelecem *arrays* multidimensionais (*tensors*) entre os nós e a comunicação distribui-se entre os vários sub-processamentos [84]. A *TensorFlow* foi desenvolvida como um melhoramento do sistema anterior da Google, o *DistBelief*, tentando superar algumas das suas limitações. É muito usada para reconhecimento de voz, procura por voz, análise de sentimento e deteção de falhas.

3.5.2. *Keras*

A *Keras* é uma biblioteca *open source* de redes neuronais escrita em Python[16] que corre por cima de, por exemplo, a *TensorFlow*, a *Theano* e outros sistemas. O propósito desta biblioteca consistiu em acelerar os treinos com redes de *deep learning* [86].

É um sistema que se comporta como uma interface/API e não um sistema *standalone*. Ela é considerada como *user-friendly*, modular e extensível, possibilitando uma abstração mais intuitiva e facilitando o desenvolvimento de modelos de redes *deep learning* independentemente do *backend* utilizado para processamento [85].

Esta API pode ser utilizada com redes neuronais tradicionais, com redes recorrentes ou com combinações de ambas. A *Keras* concebe os seus modelos como módulos *standalone* que podem ser combinados para criar novos modelos. Assim, camadas de redes neuronais, funções de custo, inicializadores de esquemas, funções de ativação, etc. constituem módulos que podem ser combinados para originar novos modelos [86].

[16] https://www.python.org

3.5.3. *Apache Mahout*

O *Apache Mahout* foi desenvolvido pela *Apache Software Foundation*[17] para criar implementações livres de algoritmos de ML escaláveis e distribuídos [87]. Tais algoritmos são destinados a tarefas de *clustering*, classificação e filtragem colaborativa. Muitas destas implementações utilizam o *Apache Hadoop* que descreveremos no tópico *3.6.5*. O objetivo desta ferramenta é o de proporcionar a construção de aplicações inteligentes de forma fácil e rápida [87].

3.5.4. *Scikit-learn*

A *Scikit-learn* é uma *framework* de ML que fornece um conjunto de ferramentas de DM e de análise de dados. É uma ferramenta *open source* construída em *NumPy*[18], *SciPy*[19] e *Matplotlib*[20] [88]. Permite executar tarefas de classificação, de regressão, de *clustering*, de redução de dimensões, de seleção de modelos e de pré-processamento. As tarefas de classificação são aplicadas ao reconhecimento de imagem e à deteção de *spam* e utilizam algoritmos como o SVM, o *k-nn,* entre outros. A regressão é aplicada para a predição de valores contínuos associados como por exemplo no estudo da resposta do organismo a medicamentos. O *clustering*, utilizando algoritmos como o *K-Means*, é aplicado a tarefas de categorização de clientes, por exemplo. A redução de dimensões pretende encurtar o número de variáveis *random* a ter em conta e é aplicada a tarefas de visualização e de aumento de eficiência. A seleção de modelos permite comparar, validar e escolher parâmetros e modelos e tem como objetivo melhorar a precisão através do ajuste de parâmetros. Alguns módulos utilizados para esta tarefa são a pesquisa por *grids* e a validação cruzada. O *Scikit-learn* providencia ainda uma funcionalidade de pré-processamento que permite a extração e normalização dos dados com aplicações, por exemplo, na transformação de dados de texto para usar em algoritmos de ML [87].

3.5.5. *Apache Hadoop*

A *framework Apache Hadoop*[21] é uma biblioteca vocacionada para o processamento de grandes quantidades de dados distribuídos por grupos de computadores, podendo esta distribuição ser feita por centenas de máquinas [89]. É uma *framework open source*

[17] https://www.apache.org
[18] https://numpy.org
[19] https://www.scipy.org
[20] https://matplotlib.org
[21] https://hadoop.apache.org

desenvolvida em *Java*[22] e altamente escalável capaz de detetar falhas na camada da aplicação. Na sua versão atual (2.9.2) a *framework* contém os módulos:

- *Common* – base de bibliotecas comuns a todos os módulos (exemplo: *Aliyn OSS Support* e *HADOOP Resource Estimater*);
- *Hadoop Distributed File System* (HDFS) – sistema responsável pelo armazenamento de dados nos clusters. É um sistema distribuído que permite elevada largura de banda.
- *Hadoop Yarn* – sistema que gere os recursos computacionais dos clusters.

O *Hadoop* implementa um paradigma computacional chamado *MapReduce* que consiste na fragmentação da aplicação em pedaços de tarefas muito pequenos cada uma das quais pode ser executada ou reexecutada em cada cluster [89].

3.5.6. *Apache Spark*

A *Apache Spark*[23] é uma *framework open source* destinada ao processamento de dados de forma distribuída sendo neste momento propriedade da *Apache Software Foundation* [90]. Permite alcançar elevada performance no processamento de dados utilizando um optimizador de *queries*, um motor de execução físico e um gerenciador *Direct Acylic Graph* (DAG) [90]. É uma *framework* de fácil utilização compatível com linguagens como Java, Scala[24], Python, R[25] e SQL. As bibliotecas providenciadas pelo *Spark* incluem: SQL e DataFrames, MLib (para ML), *GraphX* e *Spark Streaming*. Estas podem ser combinadas na mesma aplicação [90]. Uma vez que esta é uma das *frameworks* utilizadas no presente projeto, informação mais detalhada poderá ser consultada no Capítulo 4.

3.6. O *Machine Learning* aplicado à área da saúde

A aplicação dos métodos de ML à área da saúde tem tido grande popularidade nos últimos anos e o interesse neste tema tem vindo a crescer [91]. Algumas dessas aplicações incluem:

- Diagnóstico em imagiologia médica – o *deep learning* tem assumido um papel cada vez mais preponderante nas técnicas de diagnóstico, sobretudo, através de imagem fornecendo mais fontes de dados (exemplo: formas mais ricas e mais diversificadas de imagiologia) e tornando-se cada vez mais acessível. Porém, um sistema treinado com

[22] https://www.java.com
[23] https://spark.apache.org
[24] https://www.scala-lang.org
[25] https://www.r-project.org/about.html

deep learning não consegue explicar como chegou às suas predições, o que é fundamental em medicina [92].

- Aumentar a recolha de dados médicos – atualmente existe um foco em recolher dados a partir de vários dispositivos móveis de forma a agregar e estudar dados relacionados com saúde em informação utilizável. Por exemplo, a *ResearchKit*[26] da Apple visa alcançar isto para aplicar ao tratamento de doenças como Parkinson permitindo aos utilizadores o acesso a aplicações interativas com reconhecimento facial acedendo, assim, à sua condição ao longo do tempo. Com a utilização da aplicação os dados recolhidos ao longo do tempo servirão para utilizar em estudos futuros [91].
- Descoberta e fabrico de medicamentos – os métodos de ML não supervisionados têm sido utilizados para auxiliar os estudos e trabalhos que levam à descoberta de novos medicamentos [94]. Ao explorarem os dados sem supervisão podem descobrir caminhos alternativos para o tratamento de doenças multifatoriais.
- Cirurgia robótica – a cirurgia executada por robots já não é uma novidade na área da medicina. Atualmente, os robots existentes (exemplo: o Da Vinci robot[27]) são utilizados por cirurgiões em cirurgias que requerem elevados níveis de motricidade fina e detalhe em superfícies reduzidas. Para manter a estabilidade dos movimentos utiliza-se ML [93].
- Predição de surtos e epidemias – Como já foi referido no subcapítulo *2.5*, a utilização de algoritmos de aprendizagem sobre dados provenientes, por exemplo, do *social media*, tem-se prometido cada vez mais útil e eficaz na monitorização da saúde pública e deteção de epidemias em todo o mundo [93].

Atualmente, começamos a ambicionar novas formas de cuidar e investigar em saúde tirando partido das potencialidades do ML. Assim, já se começa a pensar em sistemas de medicina personalizada, tratamento automático, cirurgia robótica automatizada, entre outras aplicações [90]. A utilização do ML em medicina personalizada visa conseguir, tendo em conta a história de vida do utente, a sua genética, a sua alimentação, etc., recomendar medicamentos e doses de forma personalizada tendo em conta os sintomas apresentados.

No caso dos tratamentos automáticos, pretende-se que certos tipos de tratamentos possam ser realizados sem recorrer a intervenção humana. Hakami, proprietário da *Medtronic*[28], num vídeo sobre a diabetes, considerou que o objetivo seria que as bombas de verificação de

[26] http://researchkit.org
[27] https://www.davincisurgery.com
[28] https://www.medtronic.com

insulina trabalhassem de forma autónoma para pacientes com diabetes, monitorizando o nível de açúcar no sangue e administrando insulina conforme a necessidade sem perturbar a vida do utilizador [92]. Este cenário pode ser estendido a várias outras formas de tratamento como administração de analgésicos por monitorização do nível de dor, antibióticos, a partir do rastreamento de dados relacionados com o sangue, sono, alimentação, stress, etc. [91]. A utilização do ML para automatização das cirurgias com robots é um objetivo futuro que tem merecido muita consideração pois poderia ser usado para combinar dados visuais e padrões de motricidade de forma a que os robots pudessem executar as cirurgias de forma autónoma [90].

3.7. Conclusão

Neste capítulo introduzimos os conceitos de DM e ML com especial incidência nas redes neuronais não supervisionadas visto ser um dos algoritmos relevantes no presente trabalho. A revisão teórica efetuada permite-nos concluir que o uso destas redes em diversos contextos de exploração de dados e, em particular, na área da saúde é um contributo extremamente poderoso uma vez que permite detetar padrões até então desconhecidos de forma mais rápida do que os métodos tradicionais. Foram também apresentados vários exemplos de *frameworks* existentes que auxiliam e otimizam este processo. Foi apresentada uma breve descrição da aplicação dos métodos de ML à área da saúde que demonstram o poder atual e futuro em melhorar a capacidade de investigação, tratamentos e deteção de epidemias. Em suma, a aplicação de algoritmos de ML com o uso de *frameworks* adequadas poderá contribuir muito para os avanços na área da medicina e da saúde pública.

4. Proposta

4.1. Introdução

Neste capítulo apresentamos a solução proposta para a construção de um sistema de deteção de surtos a partir da análise de dados provenientes do Twitter. A secção 4.2, visa clarificar os objetivos e motivações da elaboração deste trabalho. Na secção 4.3, pretende-se apresentar e explicar as tecnologias e linguagens utilizadas no projeto bem como a forma como estão interligadas. Na secção 4.4 tecem-se algumas considerações finais sobre estes temas.

4.2. Solução Proposta

Sendo a análise de dados uma disciplina cada vez mais utilizadas para ajudar a tomada de decisões em várias áreas da nossa vida, considerou-se pertinente aplicar as potencialidades do ML na área da saúde. Assim, o que foi proposto desenvolver é o fornecimento de uma aplicação que permita classificar dados recolhidos do Twitter de forma a perceber se está a ocorrer um surto para uma determinada doença ou não. Pretende-se que ao utilizar esta ferramenta seja possível detetar epidemias em tempo real o que contribuirá para uma mais rápida intervenção por parte dos profissionais de saúde que poderão, por exemplo, alertar a população para a existência de um surto e, consequentemente, a adoção de estratégias de prevenção de forma mais atempada e eficaz.

4.3. Arquitetura

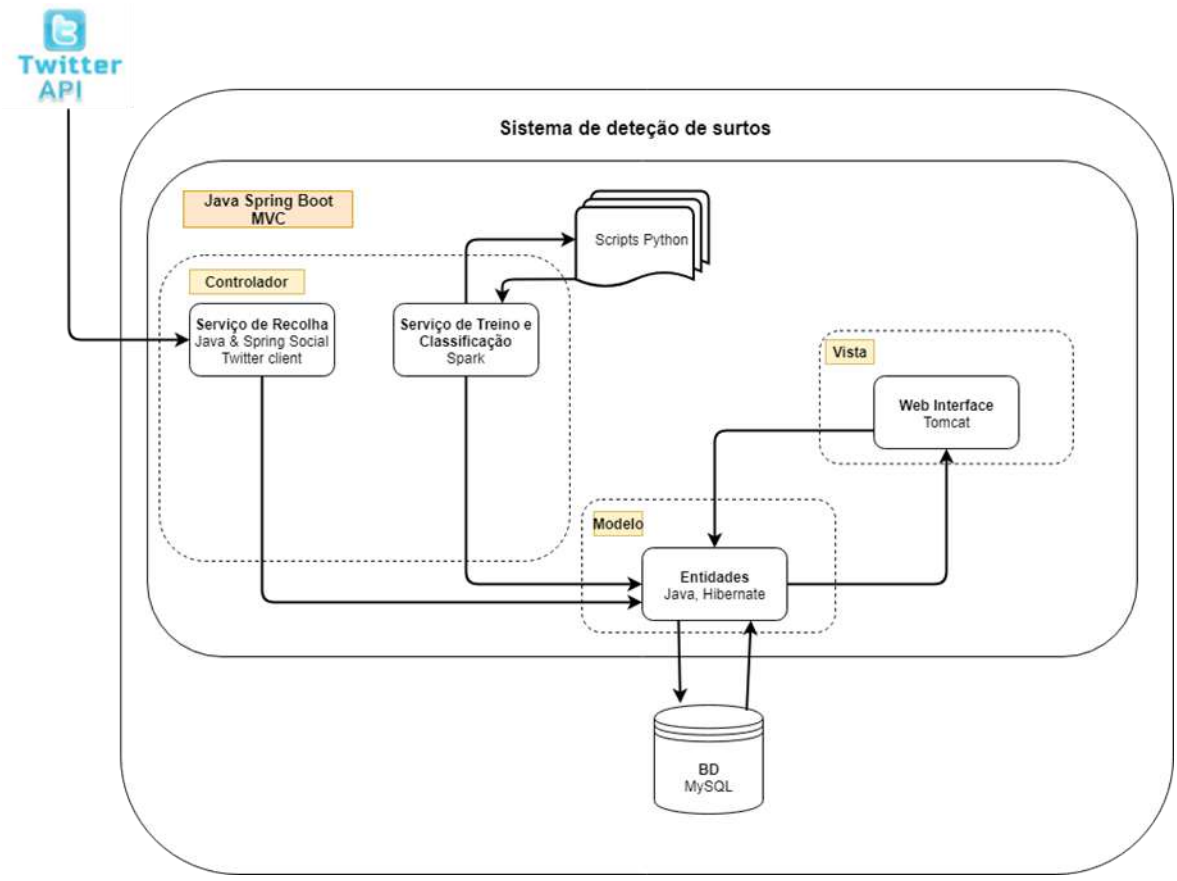

Figura 16 - Diagrama arquitetural do Sistema de deteção de surtos.

A arquitetura implementada assenta essencialmente num sistema *Java Spring Boot* [107] que utiliza o Java SE 11 com padrão arquitetural *Model-View-Controller* (MVC). É utilizado o pacote Spring Social Twitter [117] que implementa um cliente do Twitter em *Java* de forma a poder efetuar os pedidos à API do Twitter de forma mais segura e transparente. O controlador utiliza a *framework Spark* no Serviço de Treino e Classificação. Para mapear os objetos do modelo para a base de dados MySQL é utilizado um *Object-Relational Mapper* (ORM), o *Hibernate*. Posteriormente, o controlador recorre ao modelo, utilizando o ORM para mapear as entidades para a base de dados e guardar o *dataset* na base de dados MySQL. Depois, é feito o treino e construção dos modelos utilizando, novamente, as bibliotecas do *Spark-ML* e quatro scripts em *Python* que vão treinar os modelos independentemente e gerar os ficheiros com os modelos. É o controlador que dá ordem para os scripts correrem.

Os algoritmos utilizados nesta fase foram: a Regressão Logística, o *Random Forest*, Redes Neuronais Multicamadas e o *LinearSVC*. A interface de visualização de dados, construída com *Tomcat*, permite visualizar dados estatísticos referentes aos modelos treinados.

Para termos uma ideia do fluxo de execução da aplicação, inicialmente, o *Serviço de Treino e de Classificação* vai utilizar o *Spark* para treinar os modelos a partir de um dado *dataset*. Para tal, recorre aos scripts em *python*. Seguidamente, é possível recolher os dados (conjunto de *tweets* com uma determinada *hashtag*) do Twitter através da sua API utilizando o pacote Spring Social Twitter. Quem leva a cabo esta tarefa é o *Serviço de Recolha* que faz parte do controlador. Os dados recolhidos são mapeados para entidades *Java* que fazem parte do modelo e a vista apresenta os resultados da classificação. De seguida, são apresentadas as tecnologias utilizadas no projeto com maior detalhe.

4.3.1. *Spring Boot framework*

A *framework Spring Boot* para a linguagem Java permite a criação de aplicações *standalone* baseadas em *Spring* de forma intuitiva uma vez que dispensa boa parte das configurações necessárias em *Spring* (a *framework Spring* permite a utilização do padrão arquitetural MVC em *Java*) [108]. Esta *framework* possui as seguintes caraterísticas:

- Criar aplicações *Spring standalone*;
- *Tomcat* integrado;
- Fornece um conjunto de configurações opcionais para o ficheiro de dependências;
- Configuração automática das bibliotecas do Spring;
- Fornece funcionalidades prontas para produção (exemplo: métricas, healt checks e configurações externas);
- Não necessita de configuração a partir de ficheiros XML.

O *Spring Boot* corre por cima da *framerowk Spring* e utiliza um *ficheiro pom.xml* (Figura 17) para a configuração das dependências do projeto. Com base nestas configurações, o *Maven* [109] executa o *build* do projeto. O *Maven* é um software de gestão de dependências do projeto que utiliza o ficheiro "*pom"*. É o *Maven* que gere o *build* do projeto efetuando a gestão dos *plugins* a instalar no projeto.

```
1. <project>
2.   <modelVersion>4.0.0</modelVersion>
3.   <groupId>com.mycompany.app</groupId>
4.   <artifactId>my-app</artifactId>
5.   <version>1</version>
6. </project>
```

Figura 17 - Exemplo de um ficheiro *pom* [110].

4.3.2. *Spring Social Twitter*

O *Spring Social Twitter* é uma biblioteca que implementa um cliente do Twitter de forma a conseguir realizar pedidos à sua API. Trata-se de uma extensão do pacote *Spring Social* que permite a integração com o Twitter. Utiliza o *TwitterConnectionFactory* para efetuar os pedidos à API REST do Twitter.

4.3.3. Padrão *Model-View-Controller* (MVC)

O padrão MVC é um padrão arquitetural largamente usado em projetos de software devido à sua capacidade para organizar o código em componentes separados. Assim, o padrão MVC fornece três módulos concentrados nos diferentes propósitos da aplicação [110]. A Figura 18 mostra um esboço simples do fluxo MVC.

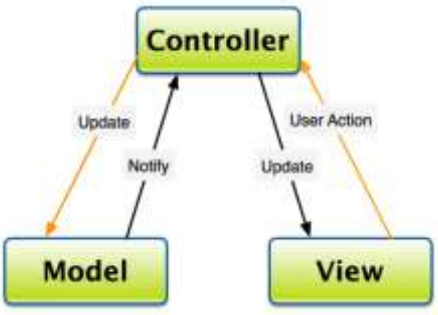

Figura 18 - Exemplo do fluxo do padrão MVC [111].

Como podemos observar na Figura 18, o MVC pressupõe a existência dos seguintes componentes:

- Do modelo – referente ao modelo de dados. O modelo representa um objeto ou uma entidade que contém ou mapeia os dados entre a base de dados e a aplicação. O modelo pode conter apenas propriedades ou ter também alguma lógica para manipulação dos dados se estes se referirem apenas ao objeto. A título de exemplo, consideremos a

entidade Estudante que tem as propriedades *nome* e *dataDeNascimento*. O modelo poderá consistir apenas na especificação destas entidades mas poderá, também, incluir métodos como *getNome()* (para ir buscar o nome do estudante), ou *setDataDeNascimento()* (para atribuir um valor à data de nascimento).

- Do controlador – é responsável por comunicar com a vista e com o modelo. O controlador executa todas as operações de lógica sobre os dados (pedindo-os ao modelo) e fornece-os à vista para serem apresentados. Portanto, é o controlador que separa a vista do modelo.
- Da vista – consiste na visualização dos dados que o modelo contém.

4.3.4. *Apache Spark*

Conforme foi referido no Capítulo 3, a *Apache Spark* é uma *framework* que disponibiliza uma API que integra com as bibliotecas do *Hadoop*. O *Spark* pode correr sobre o *Hadoop* ou de forma isolada. Este componente não tem acesso aos componentes do *Hadoop* (HFS e YARN) e por isso não tem mecanismo de persistência, pois guarda os dados em memória, tendo de recorrer a outros sistemas de armazenamento para computação distribuída (exemplo: S3[29] , Cassandra[30] ou HDFS[31]).

Uma das vantagens de utilizar o *Spark* é que esta ferramenta compila código *bytecode* (de linguagem intermédia, que pode ser proveniente de várias plataformas) para código nativo (da arquitetura da máquina) em tempo de execução permitindo melhor desempenho devido ao *Just-in-time-complier* (JIT). O *Spark* permite o processamento de dados de forma paralela e distribuída e utiliza o conceito de *DataFrame* para descrever esse conjunto de dados. Assim, a *DataFrame* corresponde a um conjunto distribuído de dados organizados em colunas disponibilizando operações de filtragem, agregações, agrupamento, entre outras. A *DataFrame* é um *dataset* organizado em colunas semelhantes a uma tabela numa base de dados relacional [107]. Estes podem ser construídos a partir de várias fontes como ficheiros de dados estruturados, bases de dados externas, etc.

A Figura 19 mostra os algoritmos de classificação e de regressão que a biblioteca do *Spark* disponibiliza.

[29] https://aws.amazon.com/pt/s3/
[30] http://cassandra.apache.org/
[31] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

Classificação	Regressão
Binomial logistic regression	*Linear regression*
Multinomial logistic regression	*Generalized linear regression*
Decision tree classifier	*Decision tree regression*
Random forest classifier	*Random forest regression*
Gradient-boosted tree classifier	*Gradient-boosted tree regression*
Multilayer perceptron classifier	*Survival regression*
Linear Support Vector Machine	*Isotonic regression*
One-vs-Rest classifier	
Naive Bayes	

Figura 19 - Algoritmos disponibilizados pela biblioteca do *Spark* [115].

Em suma, esta ferramenta foi escolhida para o processamento de dados no presente projeto devido à sua rapidez (uma vez que trabalha os dados em memória e há uma redução do número de operações de acesso ao disco para leitura e escrita), a facilidade em desenvolver aplicações paralelas, uma vez que o *Spark* fornece 80 operadores de alto nível, não haver perda de dados devido ao sistema *fault-tolerance* (que utiliza o *Spark* RDD) para lidar com falhas no cluster.

4.3.5. *Tomcat*

O *Tomcat* é um software *open source* que implementa as tecnologias *Java Servlet, JavaServer Pages* (JSP), *Java Expression Language* e *Java WebSocket* [116]. Trata-se de um servidor web para Java que contém *servlets* e tem ferramentas destinadas à configuração e gestão das páginas da interface do utilizador e que utiliza um servidor com protocolo HTTP. Também permite usar ficheiros XML para estas configurações [116]. É utilizado para renderizar páginas web utilizando código JSP.

4.3.6. *Object-Relational Mapping (Hibernate)*

O *Object-Relational Mapping* (ORM) é uma técnica utilizada para mapear os objetos do modelo para uma base de dados relacional. É responsável pelas operações de CRUD e auxilia o processo de desenvolvimento na medida em que automatiza os processos de conversão entidade-tabela e tabela-entidade, permite a utilização de menos código e um código mais limpo, melhora a performance uma vez que os objetos ficam em cache e todo o processo é transparente [112]. A título de exemplo, para a entidade estudante, o ORM terá uma tabela

estudante na base de dados onde cada coluna diz respeito às propriedades do objeto (nome, *dataDeNascimento*, etc.). O ORM também gere as relações entre as entidades. Consideremos que o estudante tem vários cursos e que um curso terá vários objetos estudante, sendo uma relação de N para N, o ORM automaticamente cria uma terceira tabela de chaves estrangeiras para representar este relacionamento [113].

No presente projeto, foi utilizado um ORM específico para Java, o *Hibernate*, que passamos agora a descrever. O *Hibernate* é uma *framework* de ORM utilizada em *Java* para mapeamento e persistência de dados aplicado a bases de dados relacionais através do *Java Database Connectiviy* (JDBC). Implementa a *Java Persistence API* (JPA) e fornece a sua própria API [113]. O *Hibernate* permite desenvolver classes persistentes seguindo a abordagem *Object Oriented Programming* (POO) utilizando os conceitos de herança, polimorfismo, associação e composição. Trata-se de uma *framework* de elevada performance e gera grande parte do SQL aquando a inicialização do sistema sendo bastante escalável e configurável [113].

4.3.7. MySQL

O MySQL é um sistema de gestão de bases de dados relacional escolhido para utilizar neste projeto [113]. É baseado na linguagem SQL (Structured Query Language) e corre em todas as plataformas. Assente numa arquitetura cliente-servidor onde o servidor se encarrega de todos os comandos da base de dados e existe separadamente do cliente para poder ser utilizado em vários ambientes. Os comandos são enviados para o servidor através do cliente que pode ser instalado no computador, por exemplo. Tem compatibilidade com inúmeros sistemas e suporta publicação em sistemas virtualizados [113]. O modelo de dados proposto para este projeto é apresentado na secção 4.3.8.

4.3.8. Modelo de Dados

O diagrama do modelo de dados utilizado neste projeto pode ser visto na Figura 20.

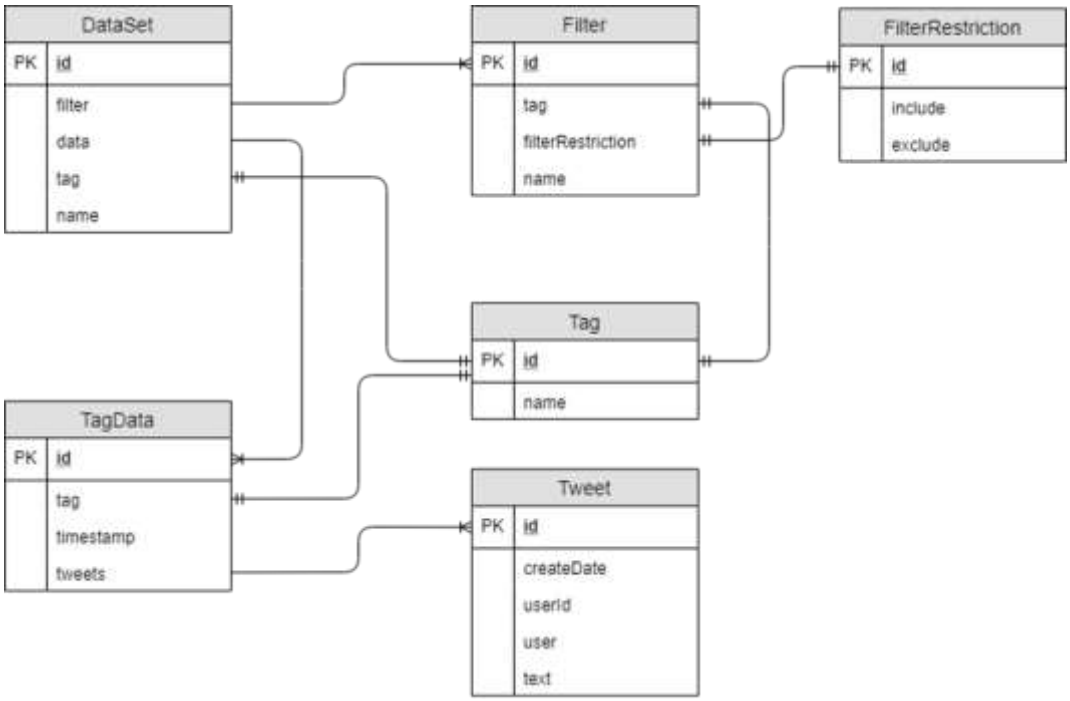

Figura 20 - Diagrama do modelo de dados do Sistema de deteção de surtos.

O modelo de dados foi concebido de forma a permitir a reutilização de *datasets* em treinos futuros bem como a classificação de *tweets* para uma determinada *tag* já existente na base de dados. Neste modelo relacional existem as entidades: *dataset*, *tagdata*, *tag*, *tweet*, *filter* e *filterrestriction*.

A entidade *dataset* é responsável por armazenar *datasets* para utilizar no treino. Esta entidade tem as seguintes propriedades: *filter* (o id do filtro a aplicar, caso exista); *data* (o id da *tagdata*); *tag* (o id da *tag*) e *name* (o nome para identificação do *dataset*). A entidade *dataset* tem uma relação de 1 para N com a entidade *filter* pois um *dataset* poderá conter uma lista de filtros a aplicar aquando do treino dos modelos. Tem também uma relação de 1 para N com a entidade *tagdata*, pois um *dataset* poderá conter várias linhas de *tags* a classificar num treino. Com a entidade tag existe uma relação de N para N onde uma *tag* poderá corresponder a vários *datasets* e vice-versa.

A entidade *tag* tem como atributos apenas o *name*, que é uma *string* indicativa do nome da *tag*. Esta entidade possui relações de N para N com as entidades *dataset*, *filter* e *tagdata*.

A entidade *tagdata* possui os campos *tag*, *timestamp* e *tweets*. A propriedade *tag* contém o id da entidade *tag* com o qual tem uma relação. O campo *timestamp* guarda a data de criação e o campo *tweets* contem o id do *tweet* ao qual está associada. A *tagdata* tem uma relação

de 1 para N com a entidade *tweet*, podendo conter uma lista de vários *tweets*, além das relações já mencionadas com as entidades *dataset* e *filter*.

O objeto *filter* contém os atributos os *tag*, *filterrstriction* e *name*, onde o primeiro remete para o id da *tag* ao qual está associado, o segundo remete para o id do objeto *filterrestriction* (com o qual mantém uma relação de N para N) e o terceiro é o nome dado ao filtro aquando a sua criação. O objeto *filterrestriction* contém as restrições que fazem parte de um dado filtro e tem duas opções: *include* e *exclude*.

Por fim, a entidade *tweet*, é onde são armazenados os *tweets* que pertencem ao objeto *tagdata* com o qual têm uma relação de N para 1. O *tweet* tem o *createdate* (a data de criação do *tweet*), o *userid* (o id do utilizador que criou o *tweet*), o *user* (o *username* do utilizador que criou o *tweet*) e o *text* (o conteúdo do *tweet*).

4.3.9. *Plotly*

O *Plotly*[32] é uma ferramenta destinada à elaboração de gráficos. Foi utilizado neste projeto para apresentar gráficos com as métricas associadas aos diferentes modelos gerados e, também, à classificação dos dados recolhidos do Twitter.

4.4. Conclusão

Neste capítulo foi apresentada a arquitetura do projeto e as tecnologias utilizadas no mesmo. Trata-se de um projeto com uma arquitetura simples tendo em vista um bom desempenho e rapidez de execução, recorrendo ao mínimo de ferramentas possível para ajudar neste processo. A base de dados e as operações de CRUD foram implementadas. No entanto, não se chegou a implementar a utilização de *datasets* da base de dados para o treino dos modelos. Desta forma, é possível criar *tags* e filtros na base de dados, mas ainda não é possível utilizar os *datasets* sendo estes atualmente fornecidos através de um ficheiro *.txt* existente numa diretoria do projeto.

[32] https://plot.ly/

5. Implementação

5.1. Introdução

Este capítulo é dedicado à descrição da implementação do projeto. Assim, será apresentada a forma como os diversos componentes do projeto foram programados e como se interligam para alcançar os objetivos propostos. A estrutura do capítulo é a seguinte: na secção *5.2*, onde é descrita a forma como o projeto *Java* se encontra estruturado, nomeadamente, em pastas; na secção *5.3* é descrita a funcionalidade que permite criar expressões de pesquisa; a secção *5.4* descreve a funcionalidade de treino dos modelos e a forma como foi implementada; a secção *5.5* apresenta a implementação da funcionalidade que permite recolher os *tweets* do Twitter e mapeá-los para entidades do modelo; a secção *5.6* explica a implementação da funcionalidade que permite classificar os *tweets* em surto ou não; por fim, a secção *5.7* descreve algumas funcionalidades que não ficaram concluídas e o que se pretende em trabalhos futuros.

5.2. Estrutura do projeto

A implementação do projeto em *Java* foi feita recorrendo a vários módulos que se encontram estruturados em diferentes pastas, são eles: o *Collector*, o *Dataclassifier* e o *Datavisualizer*. Cada uma destas pastas representa uma funcionalidade do projeto.

A pasta *Collector* contém os ficheiros responsáveis pela recolha de dados do Twitter a partir da hashtag. O módulo *Dataclassifier* é responsável pela execução do *script python* que classifica e treina o modelo. Este módulo, também devolve os resultados do treino e disponibiliza os *datasets* existentes para o treino. O módulo *Datavisualizer* é uma aplicação web que acede ao *Dataclassifier* e ao *Collector* para providenciar a interface do utilizador.

Como foi referido no capítulo 4, a configuração do projeto é efetuada mediante ficheiros *yaml*, o *application.yaml*, utilizado pelos testes, pela aplicação web e pelos submódulos. Um exemplo de um destes ficheiros de configuração pode ser consultado na Figura 21.

```yaml
local-classifier:
  interpreter: '${PYTHON_INTERPRETER}}'
  base-dir: ./classifier
  result-prefix: 'Results:'
  train-script: 'train_clf_models.py'
  classify-script: 'classify.py'


local-dataset:
    baseDir: ./classifier/datasets
```

Figura 21 - Exemplo de um ficheiro de configuração *yaml*.

5.3. Gestão de *tags*, *filtros* e *datasets*

A gestão de *tags*, filtros e *datasets* permite tornar persistente a informação relacionada com estas variáveis para ser utilizada mais tarde nos treinos e na classificação de *tweets*. Para esta implementação foi utilizado o ORM *Hibernate* e uma base de dados relacional *MySQL*. A Figura 22 mostra a página *Manage Tags* que permite definir uma determinada *hashtag* para a qual se vai mais tarde recolher dados do Twitter.

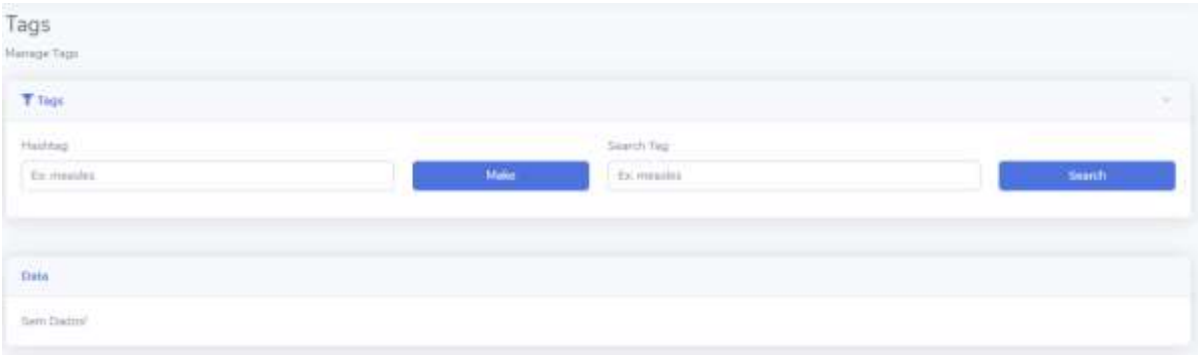

Figura 22 - Página *Manage Tags* onde é possível criar novas *tags* ou pesquisar por uma já existente.

Nesta página é possível criar uma nova *hashtag* inserindo o seu nome no campo *Hashtag* e clicando em *Make*.

A criação de *tags* é efetuada pelo controlador *TagController* através do código ilustrado na Figura 23.

```
@PostMapping("/create")
public ModelAndView create(@RequestBody TagRequest request) throws TagException
{
    ModelAndView view = new ModelAndView( viewName: "tag/index :: #searchResult");

    collectorService.createTag(request.getTagName());

    view.addObject( attributeName: "tags", collectorService.getAllTags());

    return view;
}
```

Figura 23 - Método *create()* da classe *TagController*.

O controlador recebe o nome da *tag* inserido no input da vista e recorre ao método *createTag()* da classe *CollectorServicePort* para persistir essa informação na base de dados. A vista apresenta ainda a lista de *tags* já existentes.

A página *Manage Filters*, na Figura 24, permite aplicar filtros a uma determinada *hashtag* possibilitando que, ao procurar por esta *tag* no Twitter, esses filtros sejam aplicados.

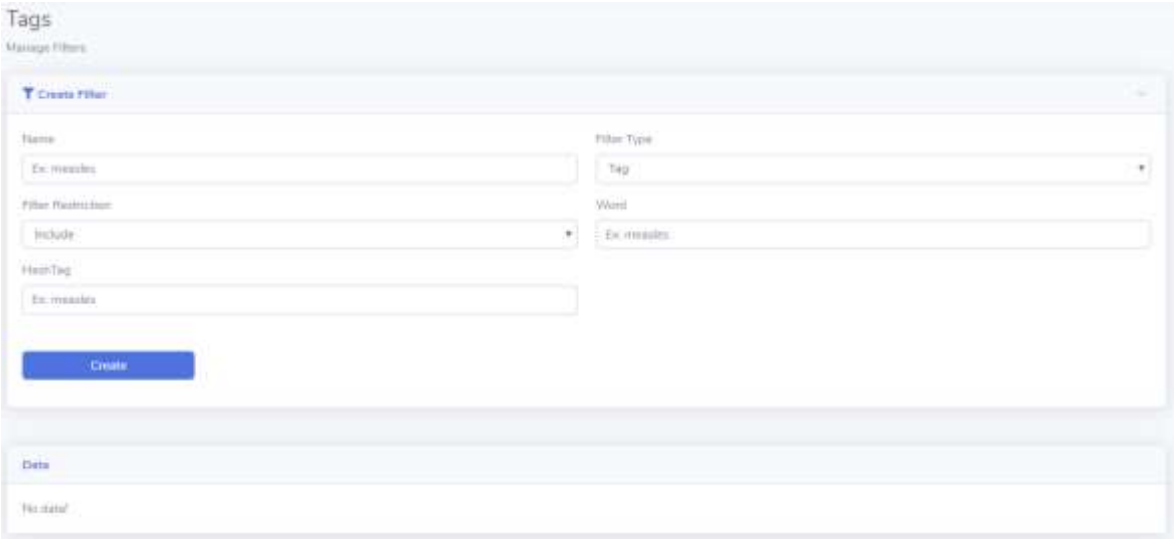

Figura 24 - Página *Manage Filters*.

O campo *Name* recebe o nome que pretendemos dar ao filtro. A *dropdowm Filter Type* permite-nos definir o tipo de filtro que queremos aplicar e assume os valores *tag* e *word*. O campo *Filter Restriction* é, também, uma *dropdown* com dois valores: *include* e *exclude*. O campo *Word* serve para indicar a palavra que pretendemos guardar no filtro (que poderá ser usada como *tag* ou como *keyword* dependendo da opção escolhida em *Filter Type* e poderá ser incluída ou excluída das pesquisas, dependendo da opção selecionada em *Filter Restriction*).

O campo *Hahstag* serve para definir a qual *tag* o filtro se irá plicar. A título de exemplo da criação de um filtro podemos considerar o seguinte cenário: para a *tag Gripe* definimos um filtro onde a palavra é *bem* e escolhemos o tipo *word* e a restrição *exclude*. Desta forma, quando pesquisarmos por *tweets* com a *tag Gripe*, irão ser excluídos os que contiverem a palavra *bem*.

A página *Manage Datasets* não chegou a ser implementada e destina-se a implementação futura.

5.4. Treino dos modelos

A *home page* (Figura 25) possui a opção *Train Classifiers* que permite treinar os classificadores do modelo. O *dataset* fornecido para o treino encontra-se na pasta */classifiers/datasets* do projeto e já possui *tweets* classificados em duas classes: 0 (não surto) e 1 (surto).

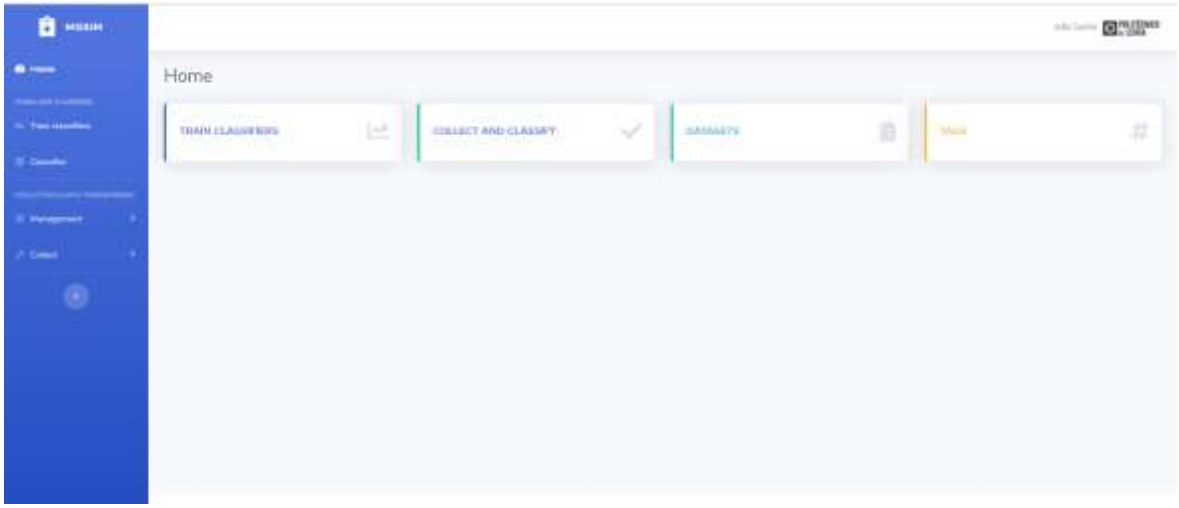

Figura 25 - *Home page* da aplicação.

Na página seguinte (Figura 26) é possível escolher um *dataset* da lista e clicar em *Train Classifiers* para efetuar o treino.

Figura 26 - Página *Classification* para efetuar o treino dos classificadores a partir do *dataset*.

Dependendo do *dataset* e dos recursos da máquina o treino poderá demorar minutos a horas para ser completado. No final do treino, é apresentada uma tabela contendo os algoritmos utilizados no treino e os valores nas métricas utilizadas. É também apresentado um gráfico com a mesma informação. Um exemplo desta vista pode ser visto na Figura 27.

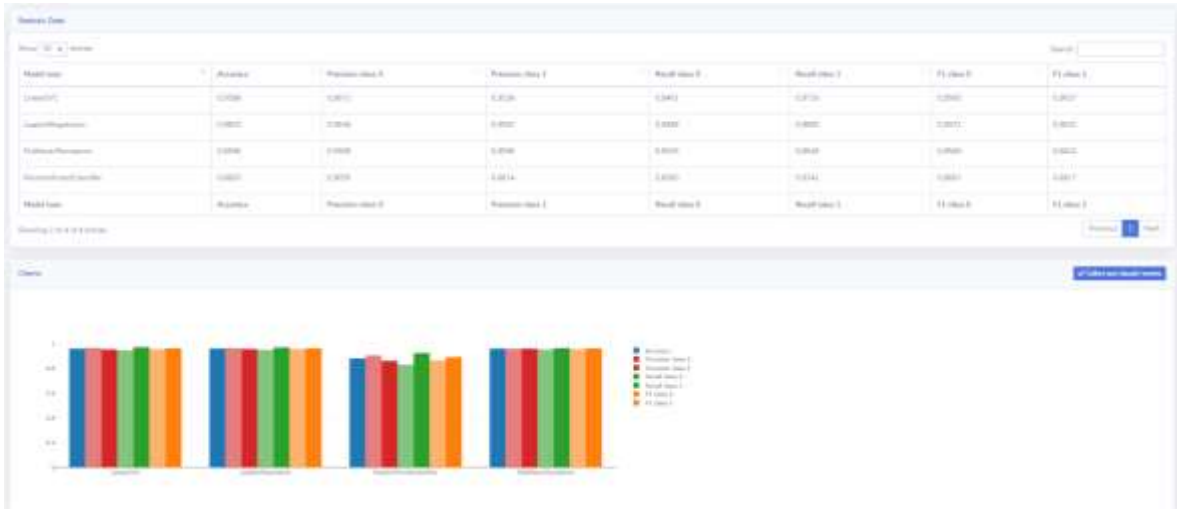

Figura 27 - Tabela e gráfico com resultados do treino.

Efetivamente, a comunicação entre esta vista e o respetivo controlador que executa a lógica pretendida é efetuada pelo módulo *datavisualizer* que contém a classe *ClassificationController*. Nela, o método *trainDataset()*, que podemos ver na Figura 28, é o responsável por construir um objeto *ModelAndView* que irá conter os resultados do treino e devolver esse objeto para ser renderizado e apresentar os resultados ao utilizador.

```
@PostMapping("/train")
public ModelAndView trainDataset(@RequestBody Dataset dataset, HttpSession session) throws ClassifierException {
    TrainingResult trainingResult = classifierService.trainClassifiers(dataset, session.getId());

    ModelAndView view = new ModelAndView( viewName: "classification/index :: #trainResult");

    session.setAttribute( s: "trainResult", trainingResult);
    view.addObject( attributeName: "trainResult", trainingResult);

    return view;
}
```

Figura 28 - Implementação do método *trainDataset()* que devolve a vista com os resultados do treino.

Os modelos são armazenados localmente na localização definida no ficheiro de configuração referido no *5.2*. Sempre que ocorre um treino, os modelos aí armazenados são substituídos pelos novos modelos. Os mesmos modelos servirão de base para classificar os *tweets* recolhidos posteriormente e identificar se se trata de um surto ou não.

O treino é efetuado com recurso à *framework Spark* e a um *script* em *python* e os algoritmos utilizados para o treino dos modelos foram: o *Random Forest; Logistic Regression, Multilayer Perceptron* e *LinearSVC*, já abordados no Capítulo 3. Cada classificador utiliza o método *cross-validation* e um conjunto de parâmetros de treino, por exemplo, número de *hidden layers* do *Multilayer Perceptron*, número de estimadores do *Random Forest*, parâmetro de penalidade de erros do *LinearSVC* e *Logistic Regression*, entre outros. O *cross-validation* consiste em dividir o *dataset* em partes de forma aleatória onde uma das partes é retirada para testes e as restantes são treinadas sendo que este processo se repete para cada conjunto dos parâmetros do modelo (mais informações no *Capítulo 6*). No final, é escolhido o melhor modelo de cada algoritmo. Daqui resultam quatro classificadores que são armazenados numa diretoria do utilizador. A configuração destes classificadores contém os *scores* atingidos para cada modelo e consiste num objeto em formato JSON que contém pares *key-value* (sendo a *key*, o modelo e o *value*, o valor).

Para cada algoritmo foram passados dois valores como input, um valor refente ao conteúdo do *tweet* e outro com a classificação (surto ou não surto). Todos os modelos devolvem apenas um valor como output que é referente à classificação do *tweet.*

Para cada um dos algoritmos foram configurados parâmetros para a execução, que são listados de seguida:

- *Multilayer Perceptron*
    - Número máximo de iterações para a solução convergir: *('max_iter': [100,200,500])*;
    - Parâmetro para regular a penalização *L2: ('alpha': [0.01, 0.001]);*
    - Número de *hidden layers: ('hidden_layer_sizes': [50,100]),* é feito uma tentativa para treinar e avaliar os modelos com 2 camadas ocultas: uma de 50 neurónios e outra com 100 neurónios;
- *Random Forest*
    - Número de *árvores: ('n_estimators': [100, 200]);*
    - Número mínimo de amostras em cada *nó folha: ('min_samples_leaf': [3, 5, 7])*;
    - Número máximo de *features* para ser usado para encontrar a solução: *('max_features': [0.5, 0.7, 0.9]);*
    - Profundidade máxima da árvore: *('max_depth': [1, 3, 5]);*
    - Número máximo de processos a correr em paralelo (usando todos os processadores): *('n_jobs': -1);*
    - Ajuste automático de pesos para a frequência das classes: *('class_weight': 'balanced');*
- *Logistic Regression*
    - Inverso da força de regularização: *('C': [0.001, 0.1, 1, 10]);*
    - Número máximo de iterações para a solução convergir: *('max_iter': [50, 100, 200]);*
    - Número máximo de processos a correr em paralelo (usando todos os processadores): *('n_jobs' = -1);*
    - Ajuste automático de pesos para a frequência das classes: *('class_weight': 'balanced');*
- *LinearSVC*
    - Parâmetro para regular a penalização por erros: *('C': [0.001, 0.1, 1]);*
    - Parâmetro para o número máximo de iterações: *('max_iter': [100, 200, 500]);*
    - Ajusto automático de pesos para a frequência das classes: *('class_weight': 'balanced');*
    - Tipo de *kernel* linear: *('kernel': 'linear');*

Todos estes parâmetros foram usados com o valor por defeito na primeira vez que foram treinados os modelos. Consoante os resultados, estes parâmetros foram ajustados para os valores que indicavam melhor performance no modelo final. Assim, a melhor configuração destes parâmetros foi mantida e usada para treinar os algoritmos.

Quanto ao vocabulário dos *tweets* utilizados também é armazenado no disco, pois será necessário para transformar os *tweets* antes do processo de classificação. Assim, pretende-se que os *tweets* tenham os mesmos campos depois do *dataset* transformado.

O módulo *Dataclassifier* possui a interface *ClarrifierService*<> que pode ser implementada dependendo do *dataset* utilizado e do tipo de utilizador para treinar modelos a partir do *dataset* e classificar uma lista de *strings*.

A classe *LocalClassifierService* implementa a interface *ClassifierService* sendo responsável pela execução dos scripts *python* com o ficheiro que contém o *dataset*. Na Figura 29 podemos ver um excerto da classe *LocalClassifierService* que contém o método *trainClassifiers()* que recebe como parâmetros um objeto do tipo *Dataset* e uma *string* referente ao utilizador.

```
@Override
public TrainingResult trainClassifiers(Dataset dataset, String user) throws ClassifierException {
    String datasetLocation = datasetDao.getDatasetLocation(dataset);

    if (datasetLocation == null) {
        throw new ClassifierException("Cannot start training: dataset does not exists");
    }

    String modelDir = FilenameUtils.concat(baseDir, user);
    return runPythonScript(datasetLocation, modelDir, MODEL_DIR_PARAM, DATASET_PARAM, trainScriptFileName, TrainingResult.class);
}
```

Figura 29 - Implementação do método *trainClassifiers()*.

Este método é responsável por aceder à localização do ficheiro *dataset* e devolve os resultados do treino que consistem num objeto do tipo *TrainingResult*. Para tal, chama o método *runPythonScript()* que vai, efetivamente executar os *scripts* de treino.

```
private <T> T runPythonScript(String dataFile, String modelDir, String modelDirParam, String dataParam,
                              String scriptFileName, Class<T> resClass) throws ClassifierException {
    new File(modelDir).mkdirs();

    String relativeScriptFileName = FilenameUtils.concat(baseDir, scriptFileName);

    String[] commandArgs = {pythonInterpreter, relativeScriptFileName, modelDirParam, modelDir, dataParam, dataFile};

    try {
        String resultString = scriptExecutor.exec(commandArgs);

        return parseResult(resultString, resultPrefix, resClass);
    } catch (IOException e) {
        throw new ClassifierException(String.format(
                "Cannot execute python script %s with commands %s.", scriptFileName, commandArgs), e);
    }
}
```

Figura 30 - Implementação do método *runPythonScript()*.

Este método é responsável por executar o *script python* que contém os algoritmos de treino para os modelos. Recebe como parâmetros: *datafile* (a localização do ficheiro que contém o *dataset*); *modelDir* (a diretoria do utilizador, onde se encontra o ficheiro); *modelDirParam* e *dataParam* que são constantes definidas nesta classe para construir o caminho para o ficheiro; *scriptFileName* (o nome do ficheiro a ler); e a classe *TrainingResult*. O método vai instanciar uma nova diretoria com o nome definido pela constante *modelDir* utilizando o construtor *File()* o método *mkdirs()*. Depois, constrói um caminho relativo para o ficheiro concatenando as *strings baseDir* e *scriptFileName*. Posteriormente, constrói um *array* de *strings* (o *commandArgs*) que contém os argumentos necessários para a execução do *script*. Seguidamente, utiliza a instância da classe *ScriptExecuter* para executar a leitura do script com o método *exec()* que recebe esses argumentos.

O resultado do método *exec()* é uma *string* com o resultado da execução que fica armazenada na variável *resultString*. Finalmente, o método *runPythonScript* devolve os resultados do treino recorrendo ao parse da *string* devolvida pelo método *exec()* que mapeia as propriedades desta *string* para um objeto do tipo *TrainingResult* e o devolve em formato JSON. O objecto *TrainingResult* tem a implementação apresentada na Figura 31 e disponibiliza métodos *get()* e *set()* para aceder às métricas: *accuracy* (fiabilidade do modelo); *precision* (precisão do modelo); *recall* (razão entre o número de verdadeiros positivos e de falsos negativos); *F1* (média pesada da *precision* e da *recall*).

```java
@Getter
@Setter
@AllArgsConstructor
public class TrainingResult {
    private Map<String, Metrics> scores;

    @Getter
    @Setter
    @AllArgsConstructor
    @NoArgsConstructor
    class Metrics {
        Double accuracy;
        Double[] precision;
        Double[] recall;
        Double[] f1;
    }
}
```

Figura 31 - Implementação do objeto *TrainingResult*.

A interface *DatasetService*, implementada pela classe *DatasetServiceImpl* fornece a lista de *datasets* disponíveis através do método *list()* presente na interface *DatasetDao*.

Depois do treino, são apresentados os resultados e gráficos relativos aos algoritmos utilizados. Assim, para cada modelo, são indicadas as seguintes métricas: o nível de *precision* de cada classe (0 e 1), a *accuracy*, o *recall* e o *F1* para cada classe.

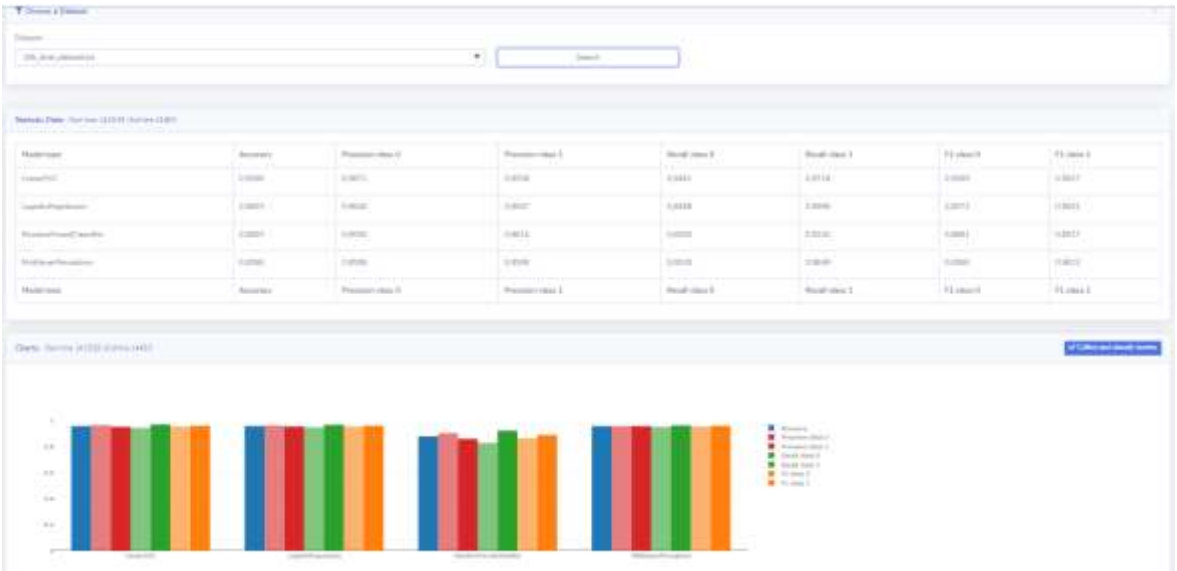

Figura 32 - Resultados do treino de classificadores para um *dataset*.

A Figura 32, mostra os resultados do treino para cada um dos algoritmos utilizados. Os resultados são mostrados na forma de tabela e de gráfico de barras onde, para cada modelo, são indicadas todas as métricas. Esta vista oferece algumas possibilidades de visualização

do gráfico a partir do conjunto de opções apresentado na Figura 33. Estas opções incluem funções de escala, *zoom*, *download* do gráfico, e visualização das métricas exatas ao fazer *hover* sobre o gráfico.

Figura 33 - Conjunto de opções de visualização do gráfico.

Depois de treinar os classificadores e construir os modelos, podemos efetuar a classificação dos dados recolhidos do Twitter.

5.5. Teste dos modelos

Após o treino, é usado o *dataset* com os restantes 20%, para ser usado no processo de teste e validação do modelo. A metodologia usada para o teste, consiste em aplicar o *dataset* de teste em cada um dos algoritmos para visualizar a sua performance com os dados previstos. Para cada um dos modelos é adicionado um histograma, como apresentado na Figura 34, com os resultados reais que cada *tweet* continha, de forma a poder visualizar a diferença dos resultados reais para as previsões de cada modelo. Será feita uma referência explicativa em tabela, na análise dos resultados.

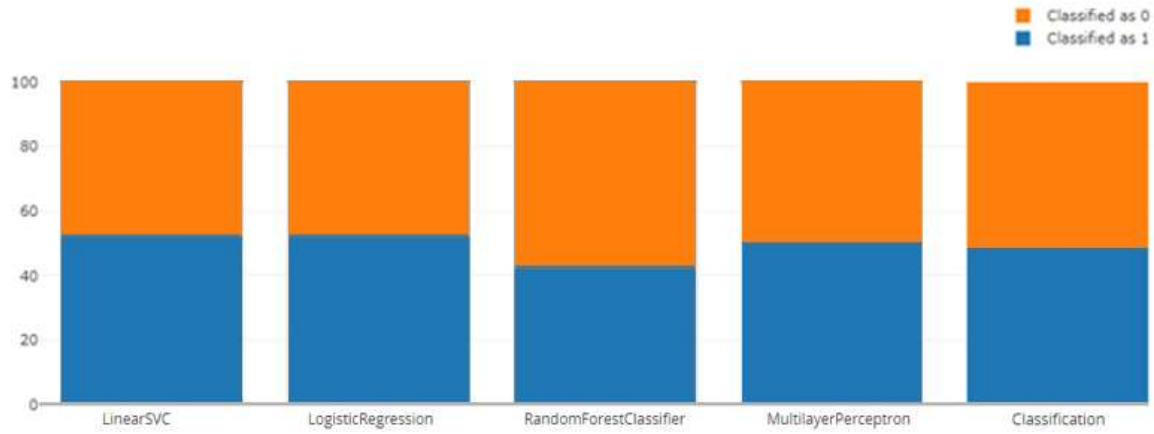

Figura 34 - Gráfico com resultados do teste.

A comunicação entre esta vista e o respetivo controlador que executa a lógica pretendida é efetuada pelo módulo *datavisualizer* que contém a classe *TestController*. Nela, o método *testDataset()*, que podemos ver na Figura 35, é o responsável por construir um objeto

ModelAndView que irá conter os resultados do teste e devolver esse objeto para ser renderizado e apresentar os resultados ao utilizador.

```
@PostMapping("/test")
public ModelAndView testDataset(@RequestBody Dataset dataset, HttpSession session) throws ClassifierException {
    TestResult testResult = classifierService.testClassifiers(dataset, session.getId());

    ModelAndView view = new ModelAndView( viewName: "classification/index :: #testResult");

    session.setAttribute( s: "testResult", testResult);
    view.addObject( attributeName: "testResult", testResult);

    return view;
}
```

Figura 35 - Implementação do método testDataset() que devolve a vista com os resultados do teste.

O código e funcionamento para o teste do modelo é idêntico ao referido no subcapítulo *5.5* no processo de treino, com a diferença de usar o restante *dataset* de 20%.

5.6. Recolha de tweets

Após treinar os modelos, a opção *Collect and classify tweets* fica disponível abaixo da tabela com as métricas dos modelos, conforme ilustra a Figura 36.

Figura 36 - Resultados das métricas dos modelos com a opção para classificação de *tweets*.

A recolha de dados a partir do Twitter é efetuada recorrendo a um cliente do Twitter para *Java* que encapsula os pedidos HTTP, o *Spring Social Twitter* (apenas necessita da configuração da API do Twitter com os dados do *consumerKey*, *consumerSecret*, *accessToken* e *accessTokenSecret*). Ao clicar em *Collect and classsify tweets* é apresentada a página *Collector* que contém um campo para escrever a *hashtag* pela qual pesquisar e um botão para iniciar a pesquisa (Figura 37).

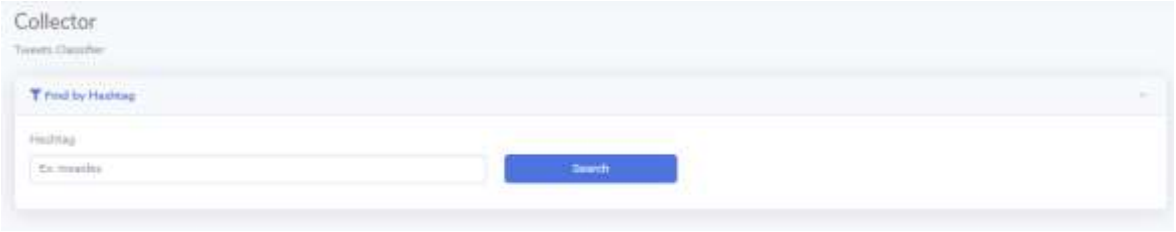

Figura 37 - Página *Collector*.

No campo *Hashtag* devemos colocar a *tag* a pesquisar no Twitter, definindo o tipo de doença cujos dados pretendemos recolher e analisar (exemplo: sarampo ou, em inglês, *measles*). Após clicar em *Search* é apresentada uma lista com os *tweets* recolhidos da API do Twitter (Figura 38) sendo possível executar duas opções: classificar todos os *tweets* (clicando no botão *Classify All*) ou classificar apenas um *tweet* (clicando em *Classify* no fim da linha do *tweet*).

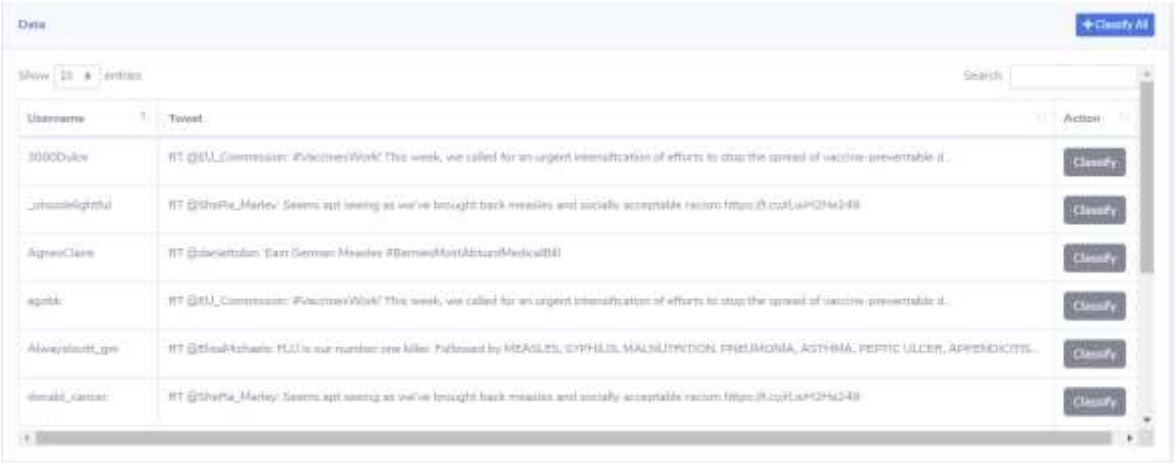

Figura 38 - Lista de *tweets* obtida a partir da API do Twitter.

A lista de *tweets* contém o valor definido no campo de pesquisa, mas também o *username*. O módulo por detrás desta funcionalidade é o *Collector* que é constituído por várias classes que passamos a explicar.

A classe *TwitterClient* é uma implementação do cliente do Twitter e é responsável pela realização do pedido HTTP que nos fornece a lista de *tweets* apresentada na Figura 38. A recolha de dados do Twitter é efetuada pelo método *searchByHashtag()* que podemos ver na Figura 39.

```
public List<Tweet> searchByHashtag(String hashtag) throws SearchParametersException
{
    if (hashtag == null || hashtag.length() <= 0)
    {
        throw new SearchParametersException("Invalid hashtag!");
    }

    return twitterTemplate.searchOperations() SearchOperations
            .search(hashtag) SearchResults
            .getTweets() List<Tweet>
            .stream() Stream<Tweet>
            .map(tweet -> TwitterUtil.getInstance().mapTweet(tweet)) Stream<Tweet>
            .collect(Collectors.toList());
}
```

Figura 39 - Implementação do método *searchByHashTag()* da classe *TwitterClient*.

Este método recebe uma *string* com o valor da *hashtag* que queremos pesquisar e devolve uma lista de *tweets*. Recorre à classe *TwitterTemplate* para a realização de operações de pesquisa (por *hashtag*) e listagem de *tweets*. O método *map()*, que recebe a função *mapTweet()* da classe *TwitterUtil* vai mapear cada *tweet* para a entidade *Tweet* e devolve uma nova *stream* com os objetos devidamente mapeados. O modelo de dados da entidade *Tweet* pode ser verificado na Figura 40.

```
public class Tweet implements Serializable
{
    private static final long serialVersionUID = 5083804874531291448L;

    @Id
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "TAG_DATA_ID")
    private TagData tagData;

    private Date createdDate;
    private long userId;
    private String user;
    private String text;

    public Tweet(String text) { this.text = text; }
}
```

Figura 40 - Implementação da classe *Tweet* com os seus parâmetros.

O objeto *Tweet* tem as propriedades: *id* (o identificador do *tweet* no Twitter), *createdDate* (data de criação), *userId* (o identificador do utilizador que criou o *tweet*), *user* (o nome do utilizador) e *text* (o texto do *tweet*).

A interface *CollectorServicePort* e a sua implementação *CollectorServicePortAdapter*, são responsáveis pela implementação do método *searchByHashtag()* que recolhe os *tweets* através da API do Twitter. Ou seja, a classe *CollectorServicePortAdapter* é responsável por chamar o método *searchByHashtag()* da instância *TwitterClient*.

5.7. Classificação de tweets

Depois de ter os modelos treinados e ter recolhido os *tweets* existem duas opções de classificação:

- *Classify* – opção presente em cada linha da lista de *tweets* que permite classificar esse *tweet* individualmente;
- *Classify All* – opção que classifica todos os *tweets* da lista.

Ao clicar em *Classify All* a lista de *tweets* é atualizada de forma a conter, para cada *tweet*, os resultados da classificação com cada modelo (Figura 41), onde:

✓ Indica que o resultado da classificação de acordo com esse modelo é positivo, ou seja, é surto (1);

✗ Indica que o resultado da classificação de acordo com esse modelo é negativo, ou seja, não é surto (0).

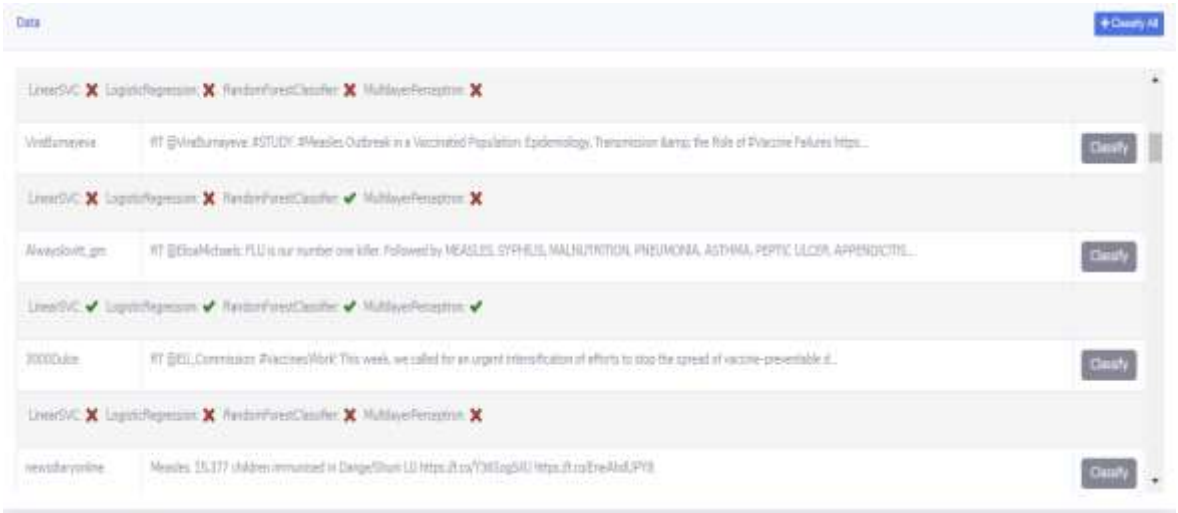

Figura 41 - Lista de *tweets* com os resultados da classificação para cada *tweet*.

Ao fazer *scroll* é, também, possível, visualizar um histograma que mostra o número de *tweets* classificados como 0 ou 1 em cada modelo, tal como se pode observar na Figura 42.

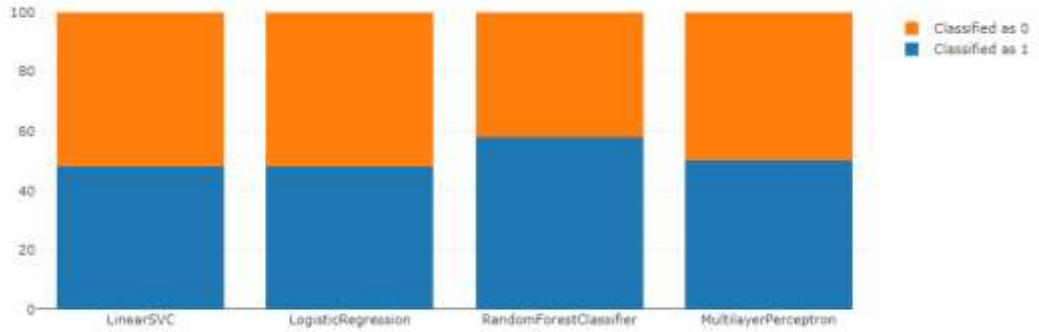

Figura 42 - Histograma com o número de *tweets* classificados com 0 e 1 para cada modelo.

A classificação dos *tweets* recolhidos é efetuada pelos ficheiros que se encontram no módulo *Dataclassifier*. Existe um *script* em *python* que efetua a classificação dos *tweets* com base nos modelos treinados. O *script* carrega os modelos a partir dos ficheiros anteriormente gerados, classifica os *tweets* e devolve os resultados para cada modelo (Figura 43). Os resultados são devolvidos num objeto em formato JSON (*predictions*) que contém pares *key-value* onde a *key* corresponde ao modelo e o *value* corresponde a um *array* com os valores da classificação (0, 1).

```
def parse_arguments():
    parser = argparse.ArgumentParser(formatter_class=RawTextHelpFormatter)
    parser.add_argument('--data', required=True, dest='data', help='A data file for prediction')
    parser.add_argument('--model_dir', required=True, dest='model_dir',
                        help='A directory where trained models are saved')
    parser.add_argument('remainders', nargs=argparse.REMAINDER)
    args = parser.parse_args()

    return args.data, args.model_dir
```

Figura 43 - Excerto do script *classify.py* em *python* utilizado para classificar os *tweets* com base nos modelos treinados.

O módulo *Dataclassifier* trabalha em conjunto com o módulo *Datavisualizer* para apresentação da interface gráfica. Este módulo contém a classe *CollectorController*

responsável por recolher os *tweets* por hashtag e utiliza o módulo *Collector* devolvendo um objeto *ModelAndView* para apresentar os *tweets*.

```
@PostMapping("/collect")
public ModelAndView collect(@RequestBody String hashtag) throws SearchParametersException
{
    ModelAndView view = new ModelAndView( viewName: "collector/index :: #searchResult");

    view.addObject( attributeName: "tweets", collectorService.searchByHashtag(hashtag));

    return view;
}
```

Figura 44 - Método *collect* da classe *CollectorController*.

Como podemos ver na Figura 44, o método *collect()* é acedido através do *endpoint /collect* quando o botão da interface *Collect Tweets* é clicado. Este método irá instanciar um novo objeto do tipo *ModelAndView* ao qual adiciona um objeto que contém uma *string* com o nome do objeto e o método que será despoletado ao clicar no botão, o *searchByHashtag()* da classe *CollectorServicePort* que integra o módulo *Collector*. Por fim, devolve a vista para ser renderizada pelo módulo *Datavisualizer*. A dinâmica é a mesma para todas as interações executadas com a interface.

O método responsável pela classificação dos resultados é o *classify()* da classe *LocalClassifierService* (a mesma utilizada para o treino dos modelos) do módulo *Dataclassifier*. A implementação deste método pode ser vista na Figura 45.

```
@Override
public ClassificationResult classify(List<String> textList, String user) throws ClassifierException {
    String modelDir = FilenameUtils.concat(baseDir, user);

    textList = textList.stream().map(s -> StringUtils.remove(s, remove: '\n')).collect(Collectors.toList());

    File dataFile = null;
    try {
        dataFile = writeTextToTempFile(textList, modelDir);

        return runPythonScript(dataFile.getAbsolutePath(), modelDir, MODEL_DIR_PARAM, DATA_PARAM,
                classifyScriptFileName, ClassificationResult.class);
    } finally {
        FileUtils.deleteQuietly(dataFile);
    }
}
```

Figura 45 - Implementação do método *classify()*.

O método *classify()* recebe uma lista de *string* com os *tweets* a classificar e uma *string* indicando o utilizador. Constrói uma *string*, a *modelDir*, por concatenação da diretoria de base (*baseDir*) e da *string user*, para armazenar os resultados. Depois, mapeia cada item da lista de *tweets* (*textList*) para uma *string* sem *'\n'*. Seguidamente, escreve essa lista em ficheiro (variável *dataFile*) e executa o método *runPythonScript()* que irá classificar os *tweets* e devolver os resultados. Para remover os ficheiros resultados do treino (contendo os modelos) quando a sessão é destruída, é utilizada a classe *SessionDestroyerListener*.

O HTML das vistas encontra-se na pasta *resources*. A pasta *templates* contém ficheiros HTML para mostrar os modelos devolvidos pelos controladores. Um exemplo deste tipo de ficheiro pode ser visto na Figura 46.

```
<!DOCTYPE html>
<tr th:if="${classificationResult  != null and classificationResult.predictions != null}"
    id="classificationResult"
    style="...">
    <td colspan="2">
        <div style="..." th:each="pred : ${classificationResult.predictions}">

            <label th:text="${pred.key + ': '}"></label>
            <label th:if="${pred.value[0] == 1}" style="...">&#10004;</label>
            <label th:if="${pred.value[0] == 0}" style="...">&#10060;</label>

        </div>
    </td>
</tr>
```

Figura 46 - Exemplo de um ficheiro HTML utilizado para mostrar os resultados da classificação

A pasta */static/js* são ficheiros *javascript* que chamam os *endpoints* da aplicação para recolher *tweets*, treinar os modelos e classificar e que atualizam os ficheiros HTML com os dados.

5.8. Conclusão

Neste capítulo apresentou-se a implementação das diferentes funcionalidades que envolvem o treino de modelos, a recolha de dados a partir do Twitter e a classificação destes dados tendo por base os modelos previamente treinados. Até ao momento, e embora já seja possível efetuar a gestão de filtros e *tags* persistindo essa informação na base de dados, ainda não é possível persistir os *datasets* nem usar a informação sobre as *hashtags* guardadas. O que se pretende, futuramente, é guardar vários *datasets* configurados e também os modelos

treinados para cada *dataset* de forma a poder escolher um deles para utilizar cada vez que se pretendem classificar *tweets*. Da mesma forma, pretende-se, ao invés de recolher dados do Twitter utilizando um campo de input onde se escreve a *tag*, ir buscar uma *tag* já existente na base de dados para pesquisar por ela, diminuindo, assim, a ocorrência de erros.

6. Análise de Funcionamento

6.1. Introdução

Neste capítulo é efetuada a análise de funcionamento. Na secção *6.2.* é feita referência à forma como os dados utilizados foram recolhidos e como deram origem aos *datasets* utilizados. Na secção *6.3* são mostrados os valores alcançados pelos diferentes algoritmos para diferentes *datasets* utilizados tendo em conta a *hashtag measles* (sarampo). Por fim, o capítulo *6.4*, remete para as conclusões da análise de resultados bem como propostas de implementação futura.

6.2. Análise dos dados recolhidos

Os dados utilizados para o treino dos modelos estão presentes num *dataset* que contém 20 000 *tweets* com a *hashtag sarampo*. Metade dos exemplos do *dataset* estão classificados como *surto* (valor 1) e os *tweets* foram recolhidos numa data em que houve ocorrência de surto. A outra metade do *dataset* está classificada como *não surto* (valor 0), em que os *tweets* foram recolhidos numa data identificada como não ocorrência de nenhum surto.

Os *tweets* classificados com valor 1 foram recolhidos quando ocorreu um surto de sarampo em 2016. Os restantes *tweets* dizem respeito a *tweets* recolhidos neste ano e não têm correspondência com nenhuma época de surto. O *dataset* utilizado inicialmente continha 10 000 *tweets*, mas apenas 200 desses *tweets* estavam classificados com valor 1 pelo que não existia equilíbrio entre as classes e os resultados eram muito inconsistentes. Assim, houve a necessidade de encontrar um *dataset* mais equilibrado. Para construir o *dataset* seguiu-se a seguinte metodologia:

- Foram efetuadas pesquisas na Internet relativas a casos de sarampo e recolheram-se as datas em que haviam ocorrido esses surtos;
- Tendo em conta a data de ocorrência dos surtos, foi efetuado *download* de 10 000 dos *tweets* existentes nessa data que contivessem a *hashtag* ou *keyword "sarampo"* (*measles*). Estes *tweets* foram categorizados com o valor 1;
- Excluindo as datas em que ocorreram as crises, foi efetuado download de 10 000 *tweets*. Estes *tweets* foram categorizados com o valor 0;
- Finalmente, os 20 000 *tweets* forma colocados num único *dataset* em ficheiro *.txt* para poderem ser utilizados pelos algoritmos de treino.

A vantagem de utilizar a API do *Spark* para treinar os modelos é que esta já possui diversas funcionalidades integradas que permitem efetuar o pré-processamento e a limpeza dos dados. Neste projeto utilizaram-se as seguintes funções de pré-processamento do *dataset*:

- Remoção de carateres ocultos e de carateres especiais, como símbolos utilizados em pontuação;
- Vectorização dos dados utilizando dos métodos *CountVectorizer()* e *fit_transform()*. Este método converte o conjunto de *strings* para uma matriz de identificadores que contém a contagem desses identificadores (onde cada identificador remete para uma *string*). Desta forma, consegue-se obter uma melhor performance na execução do algoritmo.

A Figura 45 contém um excerto do código que procede à vectorização dos dados durante a execução do programa.

```
def text_to_vector(text, model_dir):
    count_vect = CountVectorizer(max_features=10000, ngram_range=(1, 2), token_pattern='(?u)\\b\\w\\w\\w+\\b')
    X_vect = count_vect.fit_transform(text)
```

Figura 47 - Utilização do método *CountVectorizer()* para vectorização dos dados.

Como foi mencionado anteriormente, os algoritmos responsáveis pelo treino dos modelos encontram-se no ficheiro *train_clf_models.py*. Para testar a performance dos modelos foi utilizado o método de *cross-validation*. O método *cross-validation* consiste em guardar parte do *dataset* para utilizar na fase de avaliação do modelo. Existem várias formas de implementar o *cross-validation*. A mais comum, utilizada neste projeto, consistem em partir o *dataset* em *N* partes de igual tamanho onde *X* partes serão utilizadas no treino dos modelos e uma parte será utilizadas na avaliação dos modelos treinados. Desta forma, pretende-se evitar o fenómeno de *overfiting*. A pesquisa por grelha, por sua vez, pretende encontrar a melhor combinação de parâmetros para um modelo. Assim, perante vários modelos onde cada um têm uma combinação diferente dos parâmetros. Cada um destes modelos é treinado e avaliado utilizando o método de *cross-validation* para descobrir qual tem o melhor desempenho. O treino e validação dos modelos ocorreu da seguinte forma:

- Inicialmente, o *dataset* é dividido em 2 *datasets* mais pequenos, um com 80% dos dados e outro com 20%;

- Usou-se o *dataset* com 80% dos dados para treinar e validar o modelo, através da técnica de *k-fold cross validation* com 3 *folds* (visto que o *dataset* não tinha dados suficientes para fazer o treino e permitir uma boa performance dos modelos);
- O método usado foi partir o *dataset* que já continha 80% dos dados do *dataset* original, dividindo de forma aleatória em 3 partes de igual dimensão;
- Aplicou-se o *cross-validation* para cada uma das partes e para cada um dos modelos;
- O treino foi feito com dois *datasets* mais pequenos e validado com o terceiro *dataset*, repetindo o mesmo procedimento mais duas vezes alterando o *dataset* de validação;
- Foi escolhida a melhor combinação de parâmetros para o classificador e o modelo final foi treinado com esses parâmetros abrangendo 80% do *dataset*. Ou seja, 80% do *dataset* foi utilizado para efeitos de treino e 20% foi utilizado para teste e validação do modelo sendo de onde resultam as métricas utilizadas.

Pretende-se, com esta abordagem, conseguir métricas mais precisas. Mas antes, é pertinente classificar alguns conceitos relacionados com a matriz de confusão, que é frequentemente utilizada em DM para indicar os erros e acertos e baseia-se nos seguintes conceitos:

- Verdadeiros Positivos: quando a classe de positivos é corretamente classificada (o modelo classificou como surto e de facto era surto);
- Falsos Negativos: quando o modelo previu uma classe Negativo, mas o valor esperado era a classe Positivo (o modelo classificou como não surto quando na verdade era surto);
- Falsos Positivos: quando o modelo previu que a classe era positiva, mas o valor esperado era negativo (o modelo classificou como surto quando não era surto);
- Verdadeiros Negativos: quando o modelo classificou corretamente a classe negativa (o modelo classificou como não surto e de facto não era surto).

As métricas de avaliação dos resultados utilizadas neste trabalho foram:

- *Accuracy* – remete para a razão entre o número de verdadeiros positivos e o número de elementos da amostra. Indica o quão perto um valor se encontra do valor esperado. (Mede a performance do modelo treinado e mostra se o modelo foi corretamente treinado).
- *Precision* – número de verdadeiros positivos a dividir pela soma do número de falsos positivos e verdadeiros positivos. Indica a capacidade de um algoritmo identificar

corretamente os elementos de uma classe. (Ajuda a detetar quando a taxa de falsos positivos é alta).

- *Recall* – número de verdadeiros positivos a dividir pela soma de verdadeiros positivos e falsos negativos. Constitui um indicador da capacidade de um classificador para encontrar todas as amostras positivas sendo que o melhor valor possível é 1 e o pior é 0. Ou seja, indica a capacidade de um classificador conseguir identificar os casos de verdadeiros positivos. (Ajuda a detetar quando a taxa de falsos negativos é alta).
- *F1* – razão entre a precisão e o *recall*. Constitui um bom indicador de performance. O melhor resultado é 1 e o pior resultado é 0. (Mede uma performance geral da precisão do modelo através da combinação do Precision e do Recall).

A Figura 48 seguinte ilustra as fórmulas das diferentes métricas utilizadas para a análise de desempenho.

Métrica	Fórmula
Accuracy	$\frac{VP + VN}{VP + VN + FP + FN}$
Precision	$\frac{VP}{VP + FP}$
Recall	$\frac{VP}{VP + FN}$
F1	$\frac{2 * Precision * Recall}{Precision + Recall}$

Figura 48 - Tabela com as métricas utilizadas neste estudo.

Os testes foram realizados utilizando a API do *Spark*. Os parâmetros foram aplicados sobre todos os modelos resultantes dos quatro algoritmos utilizados: *Random Forest, LinearSVC, Multilayer Perceptron* e *Logistic Regression*. Deste modo é possível obter a comparação de desempenho entre os diferentes modelos e escolher o que se adapta melhor ao caso em estudo.

6.3. Análise dos resultados

Como foi referido anteriormente, inicialmente foi utilizado para treino dos modelos um *dataset* com 10 000 *tweets* dos quais apenas 1,6% constituíam casos de surto. Esta situação originou os resultados que se podem observar na Figura 49, referentes à avaliação do desempenho de cada classe em separado e para cada métrica.

Model type	Accuracy	Precision class 0	Precision class 1	Recall class 0	Recall class 1	F1 class 0	F1 class 1
LinearSVC	0.9737	0.9824	0.2245	0.9859	0.4231	0.9866	0.2933
LogisticRegression	0.9742	0.9839	0.2593	0.9799	0.5385	0.9668	0.3500
RandomForestClassifier	0.7968	0.9969	0.0494	0.7967	0.8077	0.8856	0.0931
MultilayerPerceptron	0.9851	0.9900	0.3750	0.9950	0.2308	0.9925	0.2857
Model type	Accuracy	Precision class 0	Precision class 1	Recall class 0	Recall class 1	F1 class 0	F1 class 1

Figura 49 - Resultados do treino para o primeiro *dataset*.

Como se pode observar, os resultados de *precision* para a classe 1 na maioria dos algoritmos são muito baixos quando o *dataset* não tem um número equilibrado de *tweets* distribuídos por ambas as classes. Como apenas 1,6% do *dataset* contém a classificação 1, isto acaba por influenciar negativamente os valores de *precision* para esta classe (o valor mais alto que se conseguiu foi 0,375, com o algoritmo *Multilayer Perceptron*), significando que nenhum dos modelos conseguiu classificar corretamente os *tweets* desta classe, por oposição ao que ocorreu com a *precision* da classe 1 que atinge valores máximos em todos os algoritmos. À semelhança do que ocorreu com a métrica *precision*, também os valores de *recall* para a classe 0 são satisfatórios em todos os algoritmos, à exceção do *Random Forest* que apenas obteve 0,796; mas para a classe 1, todos os algoritmos voltam a apresentar um mau desempenho com resultados abaixo dos 50% para todos os algoritmos, exceto o *Random Forest* que consegue alcançar os 80% que, ainda assim, não é um ótimo resultado. Estes dados sugerem que os modelos apresentam dificuldades em encontrar resultados relevantes para a classe 1, o que vai contra os objetivos deste projeto que se pretendem identificar os casos de surto de doenças.

Para a métrica *F1*, voltam a verificar-se os mesmos bons resultados para a classe 0 e o mau desempenho dos modelos no que toca à classe 1. Estes resultados são o que se esperaria tendo em conta o pobre desempenho dos algoritmos relativamente às medidas de *recall* e

precision para a classse 1. Com estes valores, percebemos que o *dataset* não poderia ser constituído maioritariamente por elementos de uma classe e que era necessário encontrar mais *tweets* que pudessem ser classificados com valor 1, de forma a treinar os modelos e alcançar um melhor desempenho.

Concluiu-se que estes dados não tinham qualidade suficiente para construir um modelo que permitisse detetar com precisão um surto de sarampo, pelo que foi necessário construir um novo *dataset* com maior número de *tweets* e, também, uma distribuição mais igualitária dos *tweets* pelas duas classes. Sendo assim, construiu-se um *dataset* com 20 000 *tweets* onde metade estão classificados com valor 1 e metade estão classificados com valor 0.

A Figura 50 apresenta os resultados do treino para os quatro algoritmos utilizados e tendo em conta as métricas referidas anteriormente.

Model type	Accuracy	Precision class 0	Precision class 1	Recall class 0	Recall class 1	F1 class 0	F1 class 1
LinearSVC	0,9598	0,9671	0,9538	0,9461	0,9718	0,9565	0,9627
LogisticRegression	0,9603	0,9646	0,9567	0,9498	0,9695	0,9572	0,9631
RandomForestClassifier	0,8864	0,9069	0,8708	0,8432	0,9242	0,8739	0,8967
MultilayerPerceptron	0,9608	0,9637	0,9584	0,9520	0,9686	0,9578	0,9635
Model type	Accuracy	Precision class 0	Precision class 1	Recall class 0	Recall class 1	F1 class 0	F1 class 1

Figura 50 - Resultados do treino de modelos.

Os resultados para o algoritmo *LinearSVC* apresentam uma *accuracy* de 0,958 sugerindo o bom desempenho deste algoritmo em encontrar os valores esperados. A *precision* do *LinearSVC* para a classe 0 (não surto) foi de 0,967 e para a classe 1 (surto) foi de 0,9538 o que sugere que, quer para uma classe, quer para outra, este modelo consegue identificar com bastante precisão cada um dos casos, ou seja, consegue identificar cerca de 95% dos casos de surto e não surto reais.

No que concerne à medida *recall*, este modelo apresentou uma medida de 0,946 para a classe 0 e 0,971 para a classe 1 o que indica uma boa capacidade deste modelo encontrar os verdadeiros positivos, sobretudo para a classe 1. Assim, este modelo revela-se promissor para classificar corretamente os casos de surto uma vez que devolve a maioria dos resultados relevantes para esta classe. Relativamente à métrica *F1*, o *LinearSVC* apresentou valores de 0,9565 para a classe 0 e 0,962 para a classe 1. Este resultado é indicativo de que os resultados

deste algoritmo são de qualidade e encontra-se em consonância com as elevadas medidas de *precision* e *recall* obtidas pelo mesmo.

O algoritmo de *Logistic Regression* apresentou uma *accuracy* de 0,96 pelo que parece constituir, também, um bom algoritmo para classificação, conseguindo identificar corretamente 96% dos parâmetros. As medidas de *F1* para ambas as classes (0,957 para a classe 0 e 0,936 para a classe1) vêm reforçar o bom desempenho do *Logistic Regression* para utilizar como classificador quer dos casos de surto, quer dos casos de não surto). Assim, não é de estranhar que as medidas de *precision* sejam também elevadas indicando que o algoritmo conseguiu identificar 96% dos casos de não surto e 96% dos casos de surto corretamente. A *recall* para a classe 0 (0,95) e para a classe 1 (0,97) indicam muito boa capacidade deste modelo encontrar os casos relevantes para classificação.

Relativamente ao algoritmo o *Random Forest*, este apresentou uma *accuracy* de 0,89 pelo que não parece constituir um modelo tão promissor para identificar os parâmetros corretamente uma vez que cerca de 11% dos parâmetros foram incorretamente classificados. As medidas de *precision* para a classe 0 (0,91) e para a classe 1 (0,87) também sugerem menos capacidade para identificar corretamente os casos de surto e não surto. A medida *F1* (0,87 para a classe 0 e 0,90 para a classe1) é consistente com os resultados obtidos para *accuracy* e *precision* do *Random Forest*. No que concerne ao *recall*, podemos observar que para a classe 0, este algoritmo obteve um resultado de 0,84 e para a classe 1, teve um resultado de 0,92 sendo notável uma discrepância entre a capacidade para identificar corretamente os casos relevantes em cada classe. Embora pareça conseguir identificar os casos de surto parece ter mais dificuldade em conseguir o mesmo para os casos de não surto.

O algoritmo *Multilayer Perceptron* apresenta boas métricas para todas as medidas estudadas. É um algoritmo com bom nível de *accuracy*, portanto, como boa capacidade de classificar corretamente os *tweets* (com 96% dos *tweets* corretamente classificados). A *precision* do *Multilayer Perceptron* foi de 0,963 para a classe 0 e 0,958 para a classe 1, pelo que parece ser um modelo adequado para identificar corretamente os casos de surto e não surto, com uma margem de erro de apenas 5%. O *F1* de 0,957 para a classe 0 e de 0,963 para a classe 1, reforça os indícios de bom desempenho em termos de *precision* e *accuracy*. A *recall*, de 0,958 para a classe 0 e 0,952 para a classe 1, também indica boa capacidade do modelo para identificar os *tweets* relevantes quer para os casos de surto quer para os casos de não surto.

De entre os quatro algoritmos de treino utilizados, aquele que parece apresentar um pior desempenho é o *Random Forest* o que poderá explicar-se pelo facto de este algoritmo ser mais adequado para problemas que apresentam múltiplas classes e não apenas duas como é o caso do presente estudo. De facto, o *Random Forest* foi único algoritmo utilizado com resultados de desempenho abaixo dos 90% em diversas métricas como, por exemplo a *accuracy*, onde apenas chegou aos 88,6% de casos corretamente classificados. Por outro lado, o algoritmo que parece apresentar o melhor desempenho para este tipo de classificação é o *Multilayer Perceptron* com uma *accuracy* de 96% e com medidas de *precision*, *recall* e *F1* acima dos 95% em ambas as classes.

De facto, este algoritmo acabou por ter resultados muito consistentes com aqueles apresentados pelo *Logistic Regression*, onde as métricas para todas as classes diferem por poucas décimas. Por exemplo, a *accuracy* do *Logistic Regression* é de 0,9603, apenas 0,0005 menos do que o *Multilayer Perceptron*. Esta diferença mantém-se para as métricas *F1* e *precision*, existindo apenas uma diferença maior para a métrica *recall* na classe 0. Assim, o *Multilayer Perceptron* parece ser mais adequado do que o *Logistic Regression* no que concerne a identificar corretamente os casos relevantes de não surto, embora essa diferença seja bastante pequena.

O *LinearSVC* também parece ser um bom algoritmo a utilizar neste tipo de classificação, embora com alguns resultados ligeiramente inferiores aos dos algoritmos inferiores. Por exemplo, o *LinearSVC* não teve tão bom desempenho em identificar os casos relevantes para a classe de não surto (0,946) mas apresentou medidas igualmente elevadas em todas as outras métricas incluindo os melhores resultados atingidos em *recall* para a classe 1 (0,97). Ou seja, o *LinearSVC* foi o algoritmo que melhor conseguiu identificar os casos relevantes de surto de sarampo durante os treinos e apresentou medidas tão boas de *precision* e de *F1* quanto o *Multilayer Perceptron* e o *Logistic Regression* provando-se, também, um bom algoritmo que consegue identificar com precisão os casos de *surto* e *não surto*.

Importa salientar também que depois de efetuado o modelo de testes, os algoritmos que apresentaram melhor performance foram o *LinearSVC*, *Logistic Regression* e o *Multilayer Perceptron*. Sendo o algoritmo *Random Forest* o que apresentou uma diferença significativa comparado com os outros algoritmos.

A Figura 51 apresenta os resultados da classificação do *dataset* para o teste.

	Linear SVC	Logistic Regression	Random Forest	Multilayer Perceptron	Classificação Real
Classe 0 (Não surto)	8661 (86,6%)	8618 (862%)	7623 (76,2%)	8739 (87,4%)	10000 (100%)
Classe 1 (Surto)	9143 (91,8%)	9182 (91,8%)	7975 (79,8%)	9261 (92,6%)	10000 (100%)

Figura 51 - Resultados da Classificação do *dataset* de teste.

Comparando a Figura 51 com tabela apresentada na Figura 50, ou seja, referente à classificação de treino com o teste, consegue-se perceber que os dados estão corretamente classificados se analisarmos os valores de cada modelo com os das métricas, como referido anteriormente.

Também importa salientar que existiu uma diminuição na performance, visto que era um *dataset* diferente. No entanto, os modelos conseguiram ter bom desempenho com os novos dados.

A Figura 52 mostra um gráfico comparativo dos resultados obtidos em todas as classes para cada um dos algoritmos tendo em contra as métricas utilizadas.

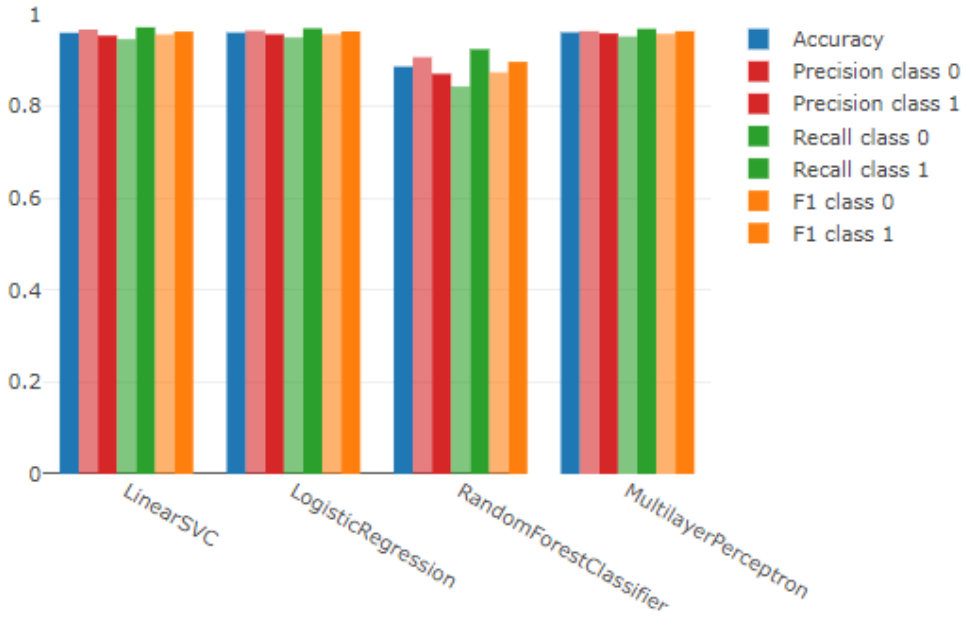

Figura 52 - Gráfico comparativo do desempenho dos diferentes algoritmos.

Pela observação do gráfico da Figura 52, é possível constatar de forma mais clara o bom desempenho, no geral, de todos os algoritmos uma vez que todos apresentaram medidas

acima dos 88% para qualquer métrica. É, também, notória a performance inferior do *Random Forest* quando comparado com os restantes três sobretudo no que concerne à medida de *recall* para a classe 0, onde apresentou os piores resultados. No geral, este algoritmo teve pior desempenho em todas as métricas.

Já para os restantes algoritmos as medidas de desempenho encontram-se muito equilibradas entre eles e qualquer um se revela um modelo de qualidade para utilizar em classificação e conseguir identificar corretamente os *tweets* relativos aos casos de *surto* de sarampo e aos casos de *não surto*.

6.4. Conclusão

Para um *dataset* com uma dimensão relevante e com dados igualmente distribuídos, todos os algoritmos utilizados apresentaram um bom desempenho e podem ser considerados de qualidade para utilizar em deteção de surtos quando a classificação é do tipo binomial. O primeiro *dataset* utilizado, além de dimensão mais reduzida, tinha uma discrepância entre as duas classes pois apenas 1,6% dos tweets eram classificados como surto, o que gerava métricas de desempenho muito pobres. Sendo, portanto, os modelos desadequados para este tipo de deteção. Um dos problemas poderá ser do *dataset* não ter dados suficientes, porque apesar de ser uma grande quantidade diversificada de *tweets*, muito provavelmente seria preciso um *dataset* com muitos *gigabytes* de informação recolhidos diariamente, onde os dados de *não surto* seriam muito superiores.

Assim, quanto à possível ocorrência de uma má classificação poderá ser devido ao *overfitting*, ou seja, mesmo que o *dataset* contenha uma grande quantidade de *tweets*, estes podem não ser suficientes para conseguir capturar todo o tipo de mensagens e vocabulário que os utilizadores poderão usar. Visto que os utilizadores podem escrever uma mensagem com o conteúdo ao seu gosto, poderá existir mensagens com conteúdo não relevante que deveria ser filtrado, mas nem sempre é possível detetar todos os casos em que isso acontece. O que poderá influenciar o resultado final quando é feita a classificação

Com isto, podemos concluir que, no que concerne à elaboração de um sistema de deteção de surtos, o cuidado com os *datasets* utilizados para o treino dos modelos é de grande importância sendo o resultado melhor quanto maior for o *dataset* e também quanto mais equilibradas estiverem as classes. Concluiu-se ainda que o algoritmo *Random Forest* não parece ser tão adequado para utilizar em casos de deteção de surtos, pelo menos, nos casos

em que apenas existem duas classes, pois este algoritmo é especificamente vocacionado para *datasets* com múltiplas classificações.

Futuramente, seria interessante utilizar mais parâmetros de análise (exemplo: localização) de forma a ter um *dataset* com mais classes e poder construir modelos com maior precisão. Talvez neste caso, o algoritmo *Random Forest* apresentasse melhores resultados. Seria, também, interessante utilizar métodos de aprendizagem não supervisionada de forma a tentar prever a ocorrência destes surtos.

7. Conclusões e trabalho futuro

O trabalho realizado mostrou ser possível construir modelos precisos capazes de detetar surtos de sarampo com dados recolhidos através do Twitter. No entanto, para tal, é necessário conseguir construir *datasets* com qualidade suficiente e com dimensão suficiente para que os modelos consigam atingir bons níveis de desempenho. Com isto se conclui que, em qualquer processo de DM, a qualidade dos dados é muito importante para alcançar os objetivos pretendidos. Durante a realização do trabalho, a API do Twitter sofreu alterações a nível comercial tornando-se impossível ir buscar *tweets* com mais de 7 dias gratuitamente, o que tem impacto ao nível da classificação dos *tweets* como é o objetivo deste trabalho.

Embora se tenha começado a implementação de um modelo de dados e da persistência dos dados em base de dados, esta funcionalidade ficou por concluir, mas seria interessante completá-la no futuro para que fosse possível guardar os *datasets* em base de dados e efetuar as pesquisas de *tweets* para classificação recorrendo a *tags* e *keywords* já previamente configuradas (quanto ao tipo de pesquisa e filtros a aplicar). Também seria interessante implementar uma funcionalidade que permitisse escolher o algoritmo a utilizar e corre-se um de cada vez, pois reduzir-se-ia consideravelmente o tempo de execução após sabermos o algoritmo que fornece o melhor modelo.

A análise de funcionamento foi efetuada para o caso específico do sarampo, mas seria possível utilizar a aplicação com qualquer outra doença desde que se construa um *dataset* adequado para treinar um modelo, onde as classes são 0 e 1. Futuramente, seria interessante ter em conta a utilização dos metadados dos *tweets* para obter informação mais precisa como, por exemplo, ter em conta a data de criação do *tweet* e a sua localização poderia ajudar a circunscrever um surto no espaço e no tempo fornecendo mais informação para a adoção de estratégias de prevenção.

Referências Bibliográficas

[1] R.D. Steele. Open source intelligent. Handbook of Intelligence Studies. 2007, p. 129 – 133.

[2] H. Woo, H. S. Cho, J. K. Lee et. al.. Identification of Keywords from Twitter and Web Blog Posts to Detect Influenza Epidemics in Korea. Disaster Medicine and Public Health Preparedness, pp. 352-359.

[3] D. Scanfeld, V. Scanfeld & E. L. Larson. Dissemination of health information through social networks: twitter and antibiotics. American Journal of Infection Control, Abril 2010; vol 38, issue 3 pp.182-188.

[4] V. Relva. A partilha de informação e aquisição de conhecimento nas redes sociais: a utilização do facebook e do google+ pelos estudantes da Faculdade de Letras da Universidade de Coimbra. Setembro 2015.

[5] J. Clement. Social Media. Statistics and Facts. [Online] https://www.statista.com/topics/1164/social-networks/. Acedido em 20 de Agosto de 2019.

[6] European Commision. Number of social media users worldwide 2010-2017 with forecasts to 2021. [Online] https://ec.europa.eu/knowledge4policy/visualisation/number-social-media-users-worldwide-2010-17-forecasts-2021_en. Acedido em 20 de agosto de 2019.

[7] D. Zeng, H. Chen, R. Lush & S. Li. Social Media Analytics and Intelligence. IEEE Computer Society. Nov/Dez 2010, pp. 1541-1672

[8] S. Jan, R. Ruby, P. T. Najeeb & M. A. Muttoo. Social Network Analysis and Data Mining. International Journal of Computer Science and Mobile Computing, vol 6, issue 6, junho 2017 pp. 401-404

[9] J. Scott. (2000). Social Network Analysis - A Handbook, Second Edition. SAGE Publication. 2000

[10] Han J, Kamber M. Data Mining: Concept and Techniques, 2nd Edition. Morgan Kauffmann. 2006

[11] Mika P. Social Networks and the Semantic Web, 1st edition. Springer, 2007.

[12] E. Culiver & M. Aufaure. Graph Mining and Communities Detection. [Online] https://hal.archives-ouvertes.fr/hal-00704356. Junho 2012. Acedido em junho de 2019

[13] E. M. Clark. Applications in sentiment analysis and machine learning for identifying public healt variables across social media. Janeiro 2019.

[14] S. Jordan et. al.. Using twitter for public health surveillance from monitoring and prediction to public response. 2019, vol 4, issue . [Online] https://www.mdpi.com/2306-5729/4/1/6/htm. Acedido a Junho de 2019

[15] J. Costa, C. Silva, M. Antunes, and B. Ribeiro. Get your jokes right: ask the crowd. Model Data Engineering, vol. 6918 LNCS, 2011 pp. 178–185.

[16] E. Jain and S. Jain. Categorizing twitter users on the basis of their interests using Hadoop/Mahout platform. 9th Int. Conf. Ind. Inf. Syst., 2014.

[17] F. Richter. Has twitter reached its natural growth limit? Statista. [Online] https://www.statista.com/chart/10460/twitter-user-growth. Acedido a maio de 2019

[18] Crannell, W. C., E. Clark, C. Jones, T. A. James, and J. Moore. A pattern-matched Twitter analysis of U.S. cancer-patient sentiments. Journal of Surgical Research, vol. 206, issue 2, 2016, pp. 536–542.

[19] M. Paul & M. Dredze. You are what you tweet: Analyzing Twitter for public health. Association for the Advancement of Artificial Intelligence. 2011 vol 20, pp. 265–272.

[20] A. Lamb, M. Paul & M. Dredze. Separating fact from fear: Tracking flu infections on Twitter. HLT-NAACL, 2013 pp. 789–795.

[21] A. Reece, A. Reagan et. al.. Forecasting the onset and course of mental illness with Twitter data. 2016. [Online] https://arxiv.org/abs/1608.07740. Acedido a Julho de 2019

[22] Paparrizos, J., White, R. W. & Horvitz, E. Screening for pancreatic adenocarcinoma using signals from web search logs: feasibility study and results. Journal of Oncologic Practice. 2016, vol 12, issue 3. [Online] https://www.ncbi.nlm.nih.gov/pubmed/27271506. Acedido a Julho de 2019

[23] E. Ferrara, O. Varol et. al.. The rise of social bots. Communications of the ACM, vol 59, issue 7, pp. 96-104

[24] M. Rodrígues-Martínez & C. Garzón-Alfonso. Twitter health surveillance (THS) system. IEEE International Conference on Big Data. 2018

[25] S. Jordan, S. Hovet, I. Fung et al. Using twitter for public health surveillance from monitoring and prediction to public response. 2018, vol 6, issue 4. [Online] www.mdpi.com/journal. Acedido em agosto de 2019

[26] J. Brownstein, C. Freifeld & L. Madoff. Digital disease detection—harnessing the web for public health surveillance. The New English Journal of Medicine. 2009, vol. 360, pp. 2153–2157

[27] S. Lewis, H. Burkom et.al.. Promising advances in surveillance technology for global health security. Disease Surveillance: Technological Contributions to Global Health Security. CRC Press: Boca Raton, 2016

[28] K. Denecke, M. Krieck et. al.. How to exploit twitter for public health monitoring. Methods of Information in Medicine. 2013, vol. 52, pp. 326–339

[29] M. Paul & M. Dredze. A model for mining public health topics from Twitter. Health. 2012, vol. 11, issue 16

[30] C. Adrover, T. Bodnar et. al... Identifying adverse effects of HIV drug treatment and associated sentiments using twitter. JMIR Public Health Surveill. 2015, vol. 1, issue 7

[31] P. Behera & S. Eluri. Analysis of public health concerns using two-step sentiment classification. International Journal of English Research in Technology. 2015, vol. 4, pp. 606–610

[32] M. Odlum & S. Yoon. What can we learn about the ebola outbreak from tweets? Infection Control Journal. 2015, vol. 43, pp. 563–571

[33] J. Gomide, A. Veloso et. Al.. Dengue surveillance based on a computational model of spatio-temporal locality of Twitter. Proceedings of the 3rd International Web Science Conference, Koblenz, Germany. 2011 pp. 15-17

[34] A. Sadilek, H. Kautz & V. Silenzio. Modeling spread of disease from social interactions. Association for the Advancement of Artificial Intelligence. 2012

[35] M. Dredze, M. Paul et. al.. Carmen: A twitter geolocation system with applications to public health. Proceedings of the AAAI Workshop on Expanding the Boundaries of Health Informatics Using AI (HIAI). 2013, pp. 20–24.

[36] A. Yepes, A. MacKinlay & B. Han. Investigating public health surveillance using Twitter. ACL-IJCNLP. 2015, vol. 164

[37] G. King, P. Lam & M. Roberts. Computer-Assisted Keyword and Document Set Discovery from Unstructured Text. American Journal of Political Science. 2017, vol. 61, pp. 971–988

[38] H. Liang & F. Shen.Privacy protection and self-disclosure across societies: A study of global Twitter users. New Media & Society. 2017, vol. 19, pp. 1476–1497

[39] L. Pollaci, A. Sîrbu et. al.. Sentiment spreading: an epidemic model for léxicon-based sentiment analysis on twiter. Springer International Publishing. 2017. [Online] https://doi.org/10.1007/978-3-319-70169-1_9. Acedido em agosto de 2019

[40] A. Khatuaa, A. Khatuac et. al.. A tale of two epidemics: Contextual Word2Vec for classifying twitter streams during outbreaks, 56, 1, 1 2019

[41] S. Lim, C. Tucker & S. Kumara. An unsupervised machine learning model for discovering latent infectious diseases using social media data. Journal of Biomedical Informatics. 2017, vol. 66, pp. 82-94

[42] S. Tuarob, S. Tucker et. al.. An ensemble heterogeneous classification methodology for discovering health-related knowledge in social media messages. Journal of Biomedical Informatics. 2014, vol. 49, pp. 255-268

[43] P. Polgreen, Y. Che net. Al.. Using influenza searches for influenza surveillance. Clinical Infectious Diseases. 2008, vol. 47, issue 11, pp.1443-1448

[44] E. Aramaki, S. Maskawa $ M. Morita. Twitter catches the flu: detecting influenza epidemics using Twitter. Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2011, pp. 1568-1576

[45] A. Sadilek, H. Kautz & V. Silenzio. Predicting Disease Transmission from Geo-Tagged Micro-Blog Data. 2012. [Online] https://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4844. Acedido em Junho de 2019

[46] C. Schmidt. Trending now: using social media to predict and track disease outbreaks. Environmental Health Perspectives. 2012, vol. 120, issue 1

[47] V. Lampos et. al.. Now casting events from the social web with statistical learning. ACM Transactions on Intelligent Technology. 2011, [Online] http://www.cs.bris.ac.uk/Publications/pub_master.jsp?id=2001449. Acedido em Agosto de 2019

[48] J. Ortiz et. al.. Monitoring influenza in the United states: a comparision of traditional surveillance systems with Google flu trends. PLoS ONE. [Online] http://dx.doi.org/10.1371/journal.pone.001868721556151. Acedido em maio de 2019.

[49] R. Zaem, D. Liau & K. Barber. Predicting disease outbreaks using social media: finding trustworthy users. Proceedings of the Future Technologies Conference. 2018, pp.369-384

[50] P. Galán-García, J. de la Puerta et. al.. Supervised machine learning for the detection of troll profiles in twitter social network: application to a real case of cyberbullying. Logic Journal of IGPL. 2015, vol. 24, issue 1, p.42-53

[51] C. MacIntyre. Social media for tracking disease outbreaks – fad or way of the future. 2016. [Online] https://theconversation.com/social-media-for-tracking-disease-outbreaks-fad-or-way-of-the-future-66401?_sm_au_=iVVN7NHZ1D0Rq4rP. Acedido em agosto de 2019

[52] A. Sadilek, H. Kautz & V. Silenzio. Modeling spread of disease from social interactions. Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media. 2012

[53] A. Alessa & M. Faezipour. A review of influenza detection and prediction through social networking sites. Theoretical Biology and Medical Modelling. 2018, vol. 15, issue 2

[54] B. Sush, L. Hong et. al.. Want to be retweeted? Large scale analytics on factos impacting retweeted in twitter network. Social Computing, IEEE Second International Conference. 2010, pp. 84-177

[55] A. Alessa & M. Faezipour. A review of influenza detection and prediction through social networking sites. Theoretical Biology and Medical Modelling. 2018, vol. 15, issue 2

[56] M. Al-garadi, M. Khan et. al.. Using online social networks to track a pandemic: a systematic review. Journal of Biomedical Informatics. 2016, vol. 62, pp. 1-11

[57] H. Carneiro & E. Mylonakis. Google trends: a web-based tool for a real-time surveillance of disease outbreaks. Clinical Infectious Diseases. 2009, vol. 49, issue 10, pp. 1557-1564

[58] J. Ginsberg et. al.. Detecting influenza epidemics using search engine query data. Nature. 2009, vol. 457, issue 7232, pp. 1012-1014

[59] A. Hulth, G. Rydevik & A. Linde. Web queries as a source for syndromic surveillance. PLoS ONE. 2009, vol. 4, issue 2

[60] M. Pennacchiotti & A. Popescu. A machine learning approach to twitter user classification. Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media. pp. 281-288

[61] C. Allen, M. Tsou et. al.. Applying GIS and machine learning methods to twitter data for multiscale surveillance of influenza, 11, 7, 7 2016

[62] X. Hu, L. Tang et. al.. Exploiting social relations for sentiment analysis in microblogging. Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. 2013, pp. 537-546

[63] Y. Yang, M. Horneffer & N. DiLisio. Mining social media and web searches for disease detection. Journal of Public Health Research. 2013. [Online] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4140326/. Acedido em julho de 2019

[64] X. Huang, M. Smith et. al.. Examining patterns of influenza vaccination in social media. [Online] https://aaai.org/ocs/index.php/WS/AAAIW17/paper/view/15200. Acedido em junho de 2019

[65] L. Charles-Smith, T. Reynolds et. al.. Using social media for actionable disease surveillance and outbreak management: a systematic literature review. 2015. [Online] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0139701. Acedido em julho de 2019.

[66] O. Missier, A. Romanovsky et. al.. Tracking dengue epidemics using twitter content classification and topic modeling. Proceedings of the 16th International Conference on Web Engineering. 2016, pp.8092

[67] C. Corley, D. Cook et. al.. Text and structural data mining of influenza mentions in web and social media. International Journal of Environmental Research and Public Health. 2010, vol. 7, issue 2, pp. 569-615

[68] D. Scanfeld, V. Scanfeld & E. Larson. Dissemination of health information through social networks: twitter and antibiotics. American Journal of Infection Control. 2010, vol. 38, issue 3, pp. 8-182

[69] K. Byrd, A. Mansuroy & O. Baysal. Mining twitter data for influenza detection and surveillance. Software Engineering in Healthcare Systems – IEEE/ACM International Workshop. 2016, pp. 9-43

[70] I. Witten, E. Frank et. al.. Data mining: pratical machine learning tools and techniques. Elsevier, 2017

[71] E. Alpaydin. Introduction to machine learning – Second Edition. Thomas Dietterich, 2010

[72] S. Pulatova & Z. Xinyou. Covering (rules-based) algorithm. [Online] https://pt.slideshare.net/totoyou/covering-rulesbased-algorithm?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em julho de 2019

[73] T. Mitchell. Machine learning. McGraw-Hill, 1997

[74] MA. Carbonneau. Introduction to multiple insatance learning. 2016. [Online] https://docplayer.net/99111276-Introduction-to-multiple-instance-learning-marc-andre-carbonneau-supervisors-eric-granger-and-ghyslain-gagnon-october-19th-2016.html. Acedido em junho de 2019

[75] Researchgate. [Online] https://www.researchgate.net/figure/An-illustration-of-the-concept-of-multiple-instance-learning-In-MIL-training-examples_fig1_315925709. Acedido em agosto de 2019

[76] C. Kumar. Machine learning types and algorithms. [Online] https://towardsdatascience.com/machine-learning-types-and-algorithms-d8b79545a6ec. Acedido em agosto de 2019

[77] Neural Networks With Java. Supervised and unsupervised leaning. [Online] https://www.nnwj.de/supervised-unsupervised.html?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em junho de 2019

[78] S. Ray. Understanding support vector machine algorithm from examples (along with code). [Online] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em maio de 2019

[79] Tutorialspoint. Data structures – divide and conquer. [Online] https://www.tutorialspoint.com/data_structures_algorithms/divide_and_conquer.htm?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em maio de 2019

[80] Skymind. A beginner's guide to neural networks and deep learning. [Online] https://skymind.ai/wiki/neural-network?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em maio de 2019

[81] Vinícius. Perceptron – redes neurais. [Online] https://www.monolitonimbus.com.br/perceptron-redes-neurais/?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em maio de 2019

[82] Matheusfacure. Redes neurais feedforward densas. [Online] https://matheusfacure.github.io/2017/05/15/deep-ff-ann/?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em junho de 2019

[83] T. Davidson. An introduction to recurrent neural networks. [Online] https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912. Acedido em maio de 2019

[84] M. Abadi, P. Barham et. al.. Tensorflow: a system for large-scale machine learning. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation. 2016

[85] A. Gulli & S. Pal. Deep learning with keras. Packt Publishing, 2017

[86] Keras. [Online] https://keras.io/?_sm_au_=iVVVW7fSj5rrGDgj#you-have-just-found-keras. Acedido em agosto de 2019

[87] G. Ingrsoll. Introducing apache mahout – scalable, comercial-friendly machine learning for buildding inteligente applications. 2009. [Online] https://www.ibm.com/developerworks/java/library/j-mahout/. Acedido em junho de 2019

[88] Scikit-learn. [Online] https://scikit-learn.org/stable/?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em junho de 2019

[89] Apache Hadoop. [Online] https://hadoop.apache.org/?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em maio de 2019

[90] Apache Spark. [Online] https://spark.apache.org/. Acedido em junho de 2019

[91] Manolis. Machine learning healthcare applications – 2018 and beyond. [Online] https://www.criticalfutureglobal.com/machine-learning-healthcare-applications-2018-and-beyond/?_sm_au_=iVVVW7fSj5rrGDgj. Acedido em maio de 2019

[92] Medtronic. Medtronic and IBM Watson Health Partner to Develop New Ways to Tackle Diabetes. [Online] https://www.medtronic.com/us-en/about/news/ibm-diabetes.html. Acedido em agosto de 2019

[93] R. Wirth & J. Hipp. CRISP-DM: towards a standard process model for data mining. Proceedings of the Fourth International Conference on The Practical Application of Knowledge Discovery and Data Mining. [Online] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.198.5133. Acedido em julho de 2019

[94] X. Zhu. Semi-Supervised Learning Literature Survey. 2005. [Online] https://minds.wisconsin.edu/handle/1793/60444?_sm_au_=iVVNN4tMt077MkW5. Acedido em maio de 2019

[95] R. Agrawal, T. Imielínski & A. Swami. Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD international conference on Management of data. 1993

[96] A. Jain, M. Murty & P. Flynn. Data clustering: a review. ACM Computing Surveys. 1999, vol. 31, issue 3, pp.264-323

[97] S. Haykin. Neural Networks and Learning Machines. Pearson Education, 2009

[98] R. Hecht-Nielsen. Theory of the Backpropagation Neural Network. [Online] https://www.sciencedirect.com/science/article/pii/B9780127412528500108. Acedido em maio de 2019

[99] X. Ma, Z. Tao et. al.. Long-short term memory neural network for traffic speed prediction using remote microwave sensor data. Transporting Research Part C: Emerging Technologies. 2015, vol. 54, pp.187-197

[100] J. Chung, C. Gulcehre et. al.. Gated feedback recurrent neural networks. Proceedings of the 32nd International Conference on Machine Learning. 2015

[101] TensorFlow. [Online] https://www.tensorflow.org/?_sm_au_=iVVJH465Z66N2WsM. Acedido em junho de 2019

[102] A. Go, R. Bhayani & L. Huang. Twitter sentimento classification using distant supervision. CS224N Project Report Stanford. 2009, vol. 1, issue 12

[103] N. Statt. How Amazon's Retail Revolçution is Changing the Way We Shop. [Online] https://www.theverge.com/2018/10/23/17970466/amazon-prime-shopping-behavior-streaming-alexa-minimum-wage. Acedido em maio de 2019

[104] G. DeAsi. How to Use Customer Behavior Data to Drive Revenue (Like Amazon, Netflix & Google). [Online] https://www.pointillist.com/blog/customer-behavior-data/?_sm_au_=iVVJH465Z66N2WsM. Acedido em maio de 2019

[105] O'Reilly. [Online] https://www.oreilly.com/library/view/distributed-computing-in/9781787126992/5fef6ce5-20d7-4d7c-93eb-7e669d48c2b4.xhtml?_sm_au_=iVVJH465Z66N2WsM. Acedido em junho de 2019.

[106] Java2Blog. [Online] https://java2blog.com/hadoop-architecture/?_sm_au_=iVVJH465Z66N2WsM. Acedido em junho de 2019

[107] Apache Spark. [Online] https://spark.apache.org/docs/latest/sql-programming-guide.html. Acedido em junho de 2019

[108] Spring Boot. [Online] https://spring.io/projects/spring-boot. Acedido em junho de 2019

[109] Apache Maven Project. [Online] https://maven.apache.org/. Acedido em maio de 2019

[110] TutorialsPoint. [Online] https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm. Acedido em maio de 2019

[111] StackOverflow. [Online] https://stackoverflow.com/questions/29594105/mvc-is-it-model-to-view-or-controller-to-view. Acedido em maio de 2019

[112] Techopedia. [Online] https://www.techopedia.com/definition/24200/object-relational-mapping--orm?_sm_au_=iVVJH465Z66N2WsM. Acedido em agosto de 2019

[113] Hibernate. [Online] https://hibernate.org/orm. Acedido em agosto de 2019

[114] MySQL. [Online] https://www.mysql.com/. Acedido em agosto de 2019

[115] J. Junceira. Análise de soluções para Dig Data Mining. 2017

[116] Apache Tomcat. [Online] http://tomcat.apache.org/. Acedido em junho de 2019

[117] Spring Social Twitter. [Online] https://docs.spring.io/spring-social-twitter/?_sm_au_=iVVJH465Z66N2WsM. Acedido em maio de 2019