INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

Análise e processamento de dados de redes sociais

CAROLINA GASPAR CÂNDIDO
(Licenciada)

Trabalho Final de Mestrado para obtenção do grau de Mestre em Engenharia Informática e de Computadores

Orientador:	Doutora Cátia Raquel Jesus Vaz
	Júri:
Presidente:	Doutor Carlos Jorge de Sousa Gonçalves
Vogais:	Doutor Artur Jorge Ferreira
	Doutora Cátia Raquel Jesus Vaz

Setembro 2020

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

Análise e processamento de dados de redes sociais

CAROLINA GASPAR CÂNDIDO
(Licenciada)

Trabalho Final de Mestrado para obtenção do grau de Mestre em Engenharia Informática e de Computadores

Orientador:	Doutora Cátia Raquel Jesus Vaz
	Júri:
Presidente:	Doutor Carlos Jorge de Sousa Gonçalves
Vogais:	Doutor Artur Jorge Ferreira
	Doutora Cátia Raquel Jesus Vaz

Setembro 2020

Agradecimentos

Aos meus pais por toda a ajuda e apoio que me deram ao longo da execução deste trabalho final de Mestrado.

Ao Nuno pela paciência e por ter sido alguém com quem pude sempre contar.

Gostaria também de agradecer à Professora Cátia Vaz pela sua orientação, ajuda, e por mostrar sempre disponibilidade para esclarecer as minhas dúvidas.

Abstract

Social media usage had a considerable increase in recent years. Twitter stands out for its ease of communication between users and for being used by the media and state figures around the world. This makes it a communication tool for spreading news around the world.

This heavy use creates a considerable amount of data, which when exploited, represents information about the opinion of its users. An example of this exploitation is that nowadays, market studies are being conducted based on the analysis of user comments on social media about a given product.

Currently, there exist many studies and tools available which can use data from Twitter to perform sentiment analysis. Whether using hashtags and emojis or using the text itself as an indicator of the polarity of the tweet. However, there is a need to use this data from Twitter to classify by the subject and not only by the tone the tweet transpires, appreciative, or negative. By classifying subjects, one can, for example, measure through the analysis of social media data regarding populism or political orientation. This analysis can be used to adapt a political campaign. However, currently, there isn't a platform that performs the various steps of this analysis.

In this work a platform that provides an analysis of tweets is elaborated. The platform relies on the framework Apache Spark for its algorithms and its ability to parallelize work. The workflow that is needed to achieve the analysis consists of obtaining the tweets, followed by filtering non relevant information and then the tweets are, in parallel, classified and organized by communities according to the relationship of influence between users established by *retweets*. This platform provides some classification algorithms such as Naïve Bayes, Random Forest and Neural Networks. It also provides some clustering algorithms such as k-Means, Gaussian Mixture and Louvain. As a result of this workflow, the platform makes available all the tweets classified, the communities found, and also presents some statistics for the data obtained. The algorithms used in classification and in clustering are chosen by the user according to the context of study. Experimental evaluations were carried out with which it was possible to observe that the best strategy, in terms of the algorithms to use, depends on the data to be analyzed.

Keywords

Machine learning, Twitter, classification, clustering.

Resumo

Nos últimos anos a utilização de redes sociais tem vindo a aumentar consideravelmente. A rede social Twitter destaca-se pela facilidade de comunicação livre entre os utilizadores e pela sua utilização por parte de figuras dos media e de estado de todo o mundo, o que faz com que seja utilizada como instrumento de comunicação sobre a atualidade de todo o mundo.

Esta forte utilização constitui uma considerável quantidade de dados, que quando explorados representam informação sobre as opiniões dos utilizadores. Um exemplo desta exploração de dados são estudos de mercados que já são realizados atualmente com base na análise dos comentários de utilizadores em rede sociais sobre um determinado produto.

Atualmente existem muitos estudos e ferramentas disponíveis para, utilizando os dados provenientes do Twitter realizar uma análise sentimental. Quer seja utilizando as *hashtags* e os *emojis* ou o próprio texto como indicador da polaridade do *tweet*. No entanto existe a necessidade da utilização destes dados provenientes do Twitter para a classificação de temas e não só pelo tom apreciativo ou negativo que o *tweet* transparece. Por classificação de temas tem-se por exemplo, aferir através da análise dos dados das redes sociais quanto ao populismo ou orientação política, podendo esta análise ser utilizada para adaptar uma campanha política. No entanto, não existe atualmente uma plataforma que realize os vários passos desta análise.

Neste trabalho foi criada uma plataforma que disponibiliza uma análise de *tweets*. A plataforma tira partido da *framework* Apache Spark para as implementações dos algoritmos, assim como da sua capacidade de paralelizar tarefas. O *worflow* que é necessário para atingir a análise de *tweets* pretendida, consiste na obtenção de *tweets*, seguida pela filtragem do seu conteúdo não relevante e, paralelamente, é realizada a classificação e o agrupamento por comunidades tendo em conta a relação de influência entre utilizadores, estabelecida pelo mecanismo de *retweet*. Como algoritmos de classificação, a plataforma tem disponíveis: *Naive Bayes*, *Random Forest* e *Neural Networks*. Como algoritmos de agrupamento, a plataforma tem disponíveis: *k-Means*, *Gaussian Mixture* e *Louvain*. Como resultado do *workflow*, a plataforma torna disponível todos os *tweets* classificados, as comunidades de utilizadores existentes e apresenta análise estatística dos resultados. Foram realizadas avaliações experimentais com as quais foi possível observar que a melhor estratégia, dos algoritmos a utilizar, depende dos dados a analisar.

Palavras-chave

Aprendizagem Automática, Twitter, classificação, agrupamento.

Índice

Abstract vii

Resumo ix

Lista de Figuras xiii

Listagens xv

Lista de Tabelas xvii

Lista de acrónimos xix

1. Introdução 1

1.1. Requisitos e funcionalidades 2
1.2. Organização do documento 5

2. Trabalho Relacionado 7

2.1. Obtenção de dados 7
2.2. Tratamento dos dados 9
2.3. Aprendizagem Automática 10
2.3.1. Aprendizagem Supervisionada 11
2.3.1.1. Naive Bayes 12
2.3.1.2. Árvore de Decisão 13
2.3.1.3. Random Forest 15
2.3.1.4. Redes neuronais 16
2.3.2. Aprendizagem não supervisionada 18
2.3.2.1. k-Means 18
2.3.2.2. Gaussian Mixture 19
2.3.2.3. Louvain 20
2.3.3. Bibliotecas e plataformas de suporte 21
2.4. Soluções Existentes 23

3. Arquitetura da solução 25

3.1. Obtenção de dados 26
3.2. Tratamento dos *tweets* 27
3.3. Classificação 27
3.4. Deteção de Comunidades 29

4. Implementação 31

4.1. Tecnologias 33
4.2. Obtenção de dados 35
4.3. Tratamento dos dados 36
4.4. Classificação 37
4.4.1. Fase de Treino 38
4.4.2. Fase de classificação 40
4.5. Deteção das comunidades 41
4.6. TwtAnalysis 44

4.6.1. Obter modelo de classificação 44
4.6.2. Efetuar predições 46

5. Avaliação experimental **49**

5.1. Metodologia 49
5.2. Conjunto de dados 49
5.2.1. Conjunto de dados do projeto HATE 50
5.2.2. Conjunto de dados *Brexit* [55] 51
5.3. Classificação 51
5.4. Deteção de comunidades 54
5.5. Conclusões experimentais 55

6. Conclusões **57**

6.1. Trabalho Futuro 57

Referências **59**

Lista de Figuras

1.1: Casos de utilização da plataforma *TwtAnalysis*. 3
1.2: *Workflow* de execução da plataforma para produzir uma análise de dados. 4
2.1: *Tweet* na plataforma Twitter. 8
2.2: Passos de criação de árvore de decisão. 15
2.3: Exemplo de rede Neuronal, designação das suas camadas [21]. 16
2.4: Operações de um neurónio [22]. 17
2.5: *Back-propagation* numa rede neuronal [23]. 17
2.6: Passos do algoritmo k-Means (a-f) [8]. 19
2.7: Passos do algoritmo *Gaussian Mixture Model* (a-f) [26]. 20
2.8: Passos do algoritmo de Louvain [27]. 21
2.9: Ecossistema Apache Spark [29]. 22
2.10: *Workflow cluster management* Apache Spark [32]. 22
3.1: Diagrama da arquitetura da plataforma desenvolvida. 26
3.2: Diagrama do módulo de tratamento de *tweets*. 27
3.3: Diagrama do módulo de Classificação dos *tweets*. 28
3.4: Diagrama do módulo de Deteção de comunidades. 29
4.1: Diagrama da solução implementada. 32
4.2: *Workflow* das etapas de tratamento de dados. 37
4.3: Módulo da classificação. 38
4.4: *Workflow* da transformação de características e categoria de um *tweet.* 39
4.5: Diagrama exemplo de agregação dos resultados de classificação e deteção de comunidades. 44
4.6: Exemplo de utilização da plataforma *TwtAnalysis* para obtenção do modelo de classificação. 46
4.7: Exemplo da utilização da plataforma *TwtAnalysis* para a obtenção de predições sobre a categorização dos *tweets.* 47
4.8: Exemplo da utilização da plataforma *TwtAnalysis* para a obtenção de predições sobre as comunidades detetadas nos *retweets.* 48
5.1: Graficos da métrica *Accuracy*. (a) gráfico do conjunto de dados Brexit. (b) gráfico do conjunto de dados HATE. 53

Listagens

2.1: *Tweet* simplificado obtido através da API do Twitter. 8
2.2: Exemplo da abordagem *bag of words* [12]..... 9
3.1: Campos relevantes da obtenção de um *tweet*. 27
4.1: Pedido HTTP à API do Twitter para obtencao de dados..... 35
4.2: Exemplo, simplificado, da reposta ao pedido da Listagem 4.1. 35
4.3: Ficheiro de *input* exemplo. 45

Lista de Tabelas

2.1: Conjunto de dados de treino do exemplo Play Tennis [18]. 12
4.1: Comparação [43] entre as linguagens de programação para a *framework* Apache Spark. 34
4.2: Parâmetros dos algoritmos de classificação. 40
4.3: Parâmetros dos algoritmos de deteção de comunidades. 42
4.4: Conjunto de dados exemplo classificados. 42
4.5: Conjunto de *retweets* dos dados exemplo classificados e agrupados. 43
5.1: Caracterização do conjunto de dados do projeto HATE. 50
5.2: Distribuição das categorias no conjunto de dados do projeto HATE. 50
5.3: Caracterização do conjunto de dados *Brexit*. 51
5.4: Distribuição de categorias do conjunto de dados *Brexit*. 51
5.5: Resultados obtidos da avaliação das técnicas de classificação, no conjunto de dados *Brexit*. 52
5.6: Resultados obtidos da avaliação das técnicas de classificação, no conjunto de dados HATE. 53
5.7: Resultados obtidos da avaliação das técnicas de agrupamento. 54

Lista de acrónimos

API	*Application Programming Interface*
CSV	*Comma-separated values*
GWT	*Google Web Toolkit*
HTTP	*Hypertext Transfer Protocol*
ID3	Induction Tree
JAR	*Java Archive*
JSF	*JavaServer Faces*
JSON	*JavaScript Object Notation*
JVM	*Java Virtual Machine*
MVC	*Model-View-Control*
REST	*Representational State Transfer*
URL	*Uniform Resource Locator*

1. Introdução

Este documento serve para descrever o projeto desenvolvido no âmbito do Mestrado em Engenharia Informática e de Computadores do Instituto Superior de Engenharia de Lisboa. O projeto desenvolvido passa pelo estudo de diferentes metodologias no tratamento e processamento de dados necessário de forma a obter uma análise sobre os mesmos. Assim, com este estudo pretende-se a realização de uma plataforma que suporte um fluxo de processamento dos dados, que inclui classificação, *clustering* e análise estatística dos dados.

Com a generalização da utilização de redes sociais tais como o Youtube [1], Facebook [2] e Twitter [3] existiu um aumento considerável de dados a serem produzidos de forma constante, dada a facilidade de estes serem produzidos através das redes sociais. Por exemplo, no caso da rede social Twitter, atualmente utilizada por figuras públicas, meios de comunicação social e figuras de Estado, é também utilizada como meio dos seus utilizadores se informarem acerca das notícias do mundo e outros temas. Podendo não só informarem-se, mas também expressarem a sua opinião livremente.

Existindo a abundância de dados provenientes de redes sociais, torna-se evidente a importância de proceder à sua obtenção e analisá-los de forma a obter um conhecimento mais profundo sobre os mesmos. É por exemplo importante em estratégias de *marketing*, que, "escutando" a perceção dos utilizadores sobre determinado produto, essa informação seja utilizada de modo a influenciá-los na compra de outros produtos. Outro exemplo importante é em épocas de eleições, para aferir a popularidade dos candidatos, averiguar a variação da intenção de voto dos eleitores consoante as ações dos candidatos ao longo da campanha eleitoral.

Uma análise sobre os dados pode ser realizada através da sua classificação, o que permite obter uma categorização dos mesmos e da aplicação de técnicas de *clustering* que permite realizar o seu agrupamento. Deste modo, será possível retirar informação e analisar fenómenos que não eram visíveis antes do seu processamento. Embora existam plataformas independentes que permitem realizar cada uma destas operações, não foi encontrada nenhuma plataforma que integrasse todas estas operações, simplificando todo o processo, nomeadamente, classificação, agrupamento e análise estatística sobre a informação adquirida.

Mais especificamente foi notada a necessidade de uma plataforma que tivesse um processo de análise semelhante, no âmbito do projeto HATE [4]. O projeto HATE passa por um estudo das atitudes e da prevalência de ódio e inverdade em debates políticos através de comentários a notícias *online*, *posts* e *tweets* [4]. Esta necessidade de estudo teve origem pois enormes quantidades de comentários, *posts* e *tweets* são publicados

todos os dias, por utilizadores de todo o mundo. Assim é útil aferir de que forma estes comportamentos dos utilizadores online afetam o debate político. O projeto HATE pretende analisar os comportamentos em diversos meios sociais online, tais como comentários em artigos de jornais, publicações em redes sociais como o Facebook, Twitter e Reddit [5]. Além de o projeto HATE poder servir de aplicação concreta neste projeto, contribui também com alguns dos dados utilizados, pois já possui um conjunto de dados classificados, e estes poderão ser utilizados para validar a plataforma desenvolvida.

Inspirado no projeto HATE, mas com vista a poder ser utilizado em outros contextos, o objetivo deste projeto de mestrado é a realização de uma plataforma de processamento e análise de *tweets*. A plataforma inclui quatro componentes fundamentais: a componente de obtenção de dados, que poderão ser obtidos através da API do Twitter ou *tweets* fornecidos pelo utilizador; a componente de tratamento de dados, que filtra conteúdo não relevante; a componente de classificação, que permite categorizar um conjunto de *tweets*; e a componente de *clustering*, que permite o agrupamento de *tweets* associados pelo mecanismo *retweet*. O conjunto destas componentes é disponibilizado seguindo um fluxo de execução das mesmas, nomeadamente: realização da obtenção dos dados, seguida do seu tratamento, os *tweets* tratados são depois utilizados para a classificação e *clustering*. Por último, combina-se a informação da classificação e do *clustering*, de forma a poder classificar os grupos existentes e produzir uma análise estatística sobre os resultados obtidos. A plataforma tem como objetivo que os utilizadores possam escolher as técnicas, disponíveis pela plataforma, que mais se adequem aos seus casos de estudo.

1.1. Requisitos e funcionalidades

Esta secção, descreve o conjunto de requisitos que se pretende que a plataforma desenvolvida no contexto deste projeto de mestrado satisfaça. Para tal, foi tido em consideração o objetivo da mesma e os casos de uso que se pretende que sejam assegurados. A plataforma *TwtAnalysis* deverá possibilitar as seguintes operações: obtenção de *tweets*; o seu pré-processamento; a sua classificação; o seu agrupamento (*clustering*) e por fim uma análise estatística sobre a informação adquirida.

No contexto da utilização da plataforma, o utilizador deve fornecer um ficheiro com um conjunto de *tweets* classificados. Este irá ser utilizado para produzir o modelo de classificação. Após a obtenção desse modelo, a plataforma também permite usar o mesmo para classificar novos *tweets*.

Como se pode observar na Figura 1.1, um caso de uso da plataforma é o treino do modelo. Para tal, o utilizador necessita de fornecer um ficheiro *Comma-Separated Values* (CSV) com um conjunto de *tweets* classificados relativamente a uma determinada característica (por exemplo, quanto ao populismo no caso de análise política dos mesmos). Isto é, uma das colunas deste ficheiro terá um cabeçalho univocamente identificado e, para cada

tweet, um valor de classificação. Após a leitura desse ficheiro, é realizado o pré-processamento do mesmo seguido da criação do modelo de classificação.

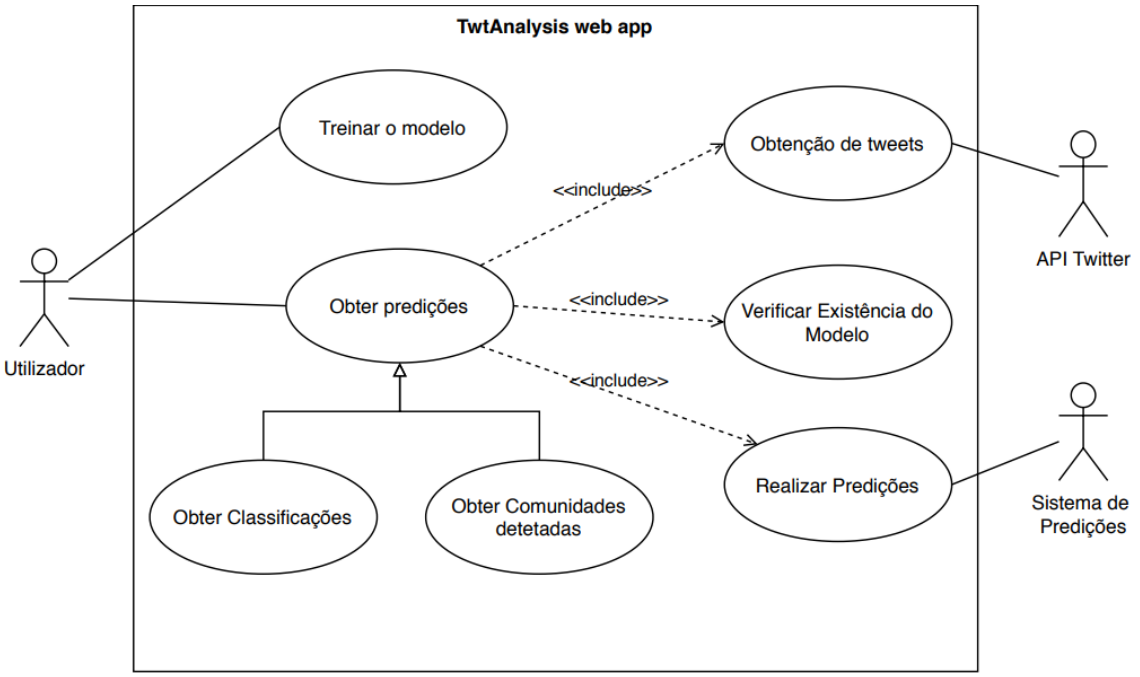


Figura 1.1: Casos de utilização da plataforma *TwtAnalysis*.

Após ter feito o treino, o utilizador pode realizar a operação "Obter Predições" (outro caso de uso da plataforma desenvolvida), que consiste na obtenção dos *tweets*, que é realizada através de pedidos à *Application Programming Interface* (API) do Twitter. Com vista a obter apenas *tweets* cujo texto tenham algum contexto para a análise pretendida, são apenas obtidos do Twitter os *tweets* que contenham um determinado conjunto de palavras chave, definidos pelo utilizador da plataforma, assim como a língua utilizada nos mesmos. Seguindo o exemplo de aferir o populismo, poder-se-ia ter como parâmetros as palavras-chave "emprego, economia" e a língua dos *tweets* "pt". Obtendo estes *tweets*, é realizado o seu tratamento, a classificação dos mesmos, e a deteção de comunidades existentes na rede construída através dos *retweets* entre utilizadores. Comparando os resultados da classificação dos *tweets* com os utilizadores a quem pertencem, é possível extrapolar esses resultados para a comunidade a que os utilizadores fazem parte e desta forma classificá-la com a categoria mais frequente. Por fim, é realizada uma análise estatística contendo todas as informações recolhidas ao longo deste caso de utilização.

Para cumprir o objetivo do projeto, assim como suportar os casos de uso previamente identificados, é necessário que a plataforma permita a execução do fluxo de etapas (*workflow*) presente na Figura 1.2, onde cada cor das setas corresponde a um fluxo diferente.

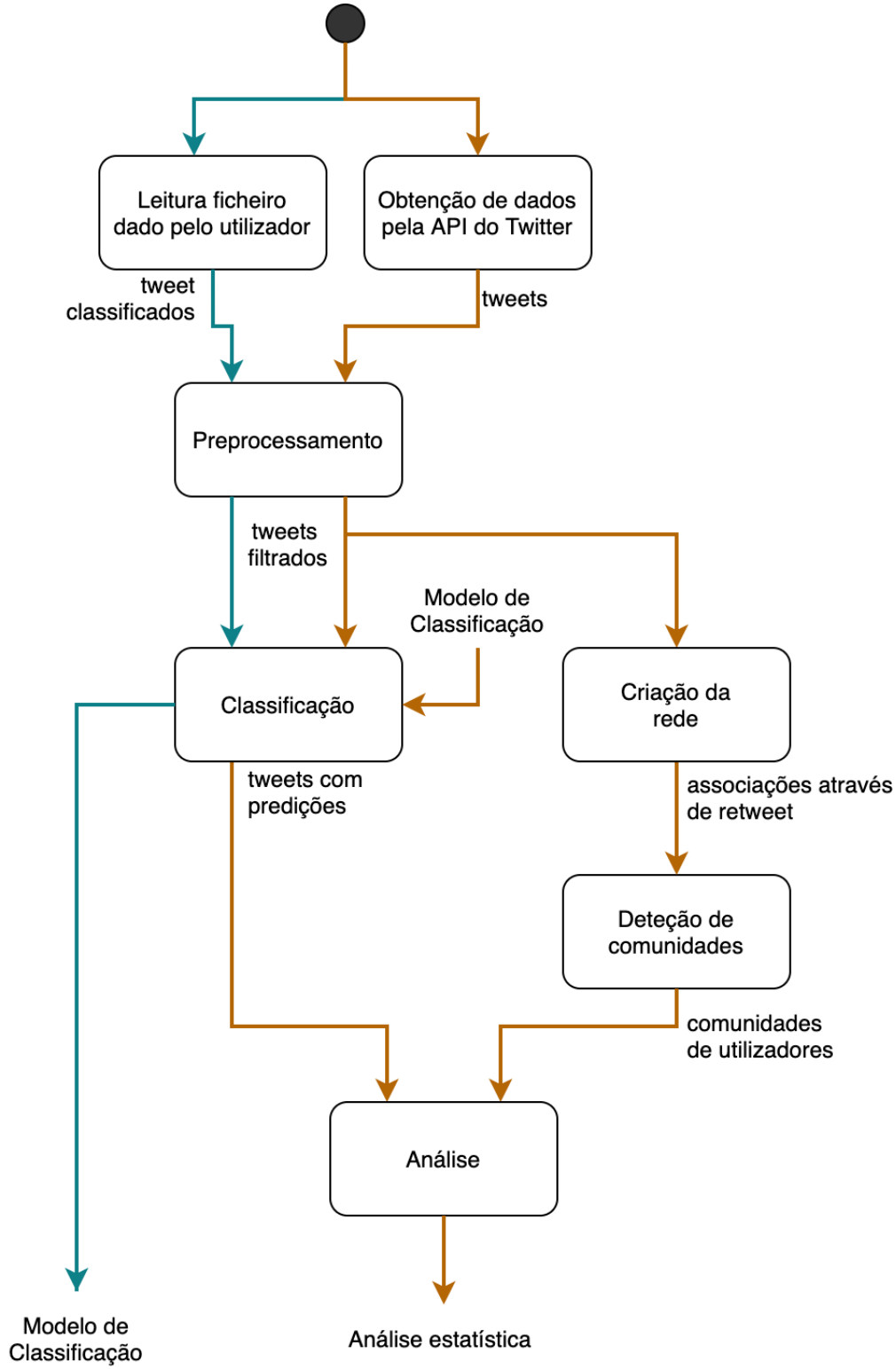


Figura 1.2: *Workflow* de execução da plataforma para produzir uma análise de dados.

De forma a poder satisfazer o *workflow* da Figura 1.2, chegou-se aos seguintes requisitos funcionais:

1. Suportar obtenção de *tweets* via ficheiro CSV facultado pelo utilizador, e proveniente da API do Twitter [6].
2. Implementar processos de processamento dos *tweets*.

3. Permitir a classificação dos *tweets* utilizando algoritmos de classificação automática, como por exemplo o algoritmo *Random Forest* [7].
4. Permitir a inferência de comunidades, utilizando algoritmos de *clustering*, como por exemplo o algoritmo *k-Means* [8].
5. Produzir uma análise estatística sobre as classificações e comunidades encontradas.

A plataforma possui dois fluxos, ilustrados na Figura 1.2 por setas de diferentes cores. O fluxo de setas de cor verde permite o treino de forma a criar um modelo de classificação. Este primeiro fluxo, setas de cor verde, consiste na leitura de um ficheiro com *tweets* previamente classificados facultado pelo utilizador, com esse ficheiro e após um pré-processamento dos *tweets* é criado, recorrendo a algoritmos de aprendizagem automática, um modelo de classificação sendo este utilizado nas predições em conjuntos de *tweets* sem classificação prévia. Este fluxo tem como *output* o modelo de classificação. Por outro lado, o fluxo de setas de cor laranja permite realizar predições sobre um conjunto de *tweets* sem classificação prévia. Este fluxo consiste na obtenção de um conjunto de *tweets* através da API do Twitter, no seu pré-processamento e utilizando o modelo de classificação, obtido no fluxo de cor verde, é possível categorizar os *tweets*. Neste fluxo é realizada a deteção das comunidades entre os utilizadores, estes são relacionados através de uma rede que considera apenas os *tweets* realizados através do mecanismo de *retweet* sendo possível inferir a forma como os utilizadores se agrupam, ou seja, as comunidades existentes. Tendo o conjunto de dados categorizados e agrupados por comunidades é possível obter resultados da distribuição das categorias dos *tweets* de um dado utilizador, assim como categorizar as comunidades encontradas, dadas as categorias mais frequentes dos utilizadores que as constituem. A plataforma produz uma análise estatística sobre os resultados obtidos que consiste nas distribuições das categorias dos *tweets* e nas distribuições das categorias existentes nas comunidades encontradas.

1.2. Organização do documento

Este documento encontra-se dividido em seis capítulos. O primeiro capítulo enquadra o problema, descreve a motivação e apresenta as funcionalidades da plataforma desenvolvida. No segundo capítulo descrevem-se possíveis abordagens a seguir nas diferentes componentes da plataforma a desenvolver. No terceiro capítulo, apresenta-se a arquitetura dos módulos da plataforma, descrevendo as suas funcionalidades, as abordagens que foram seguidas e a forma como os módulos se relacionam. No quarto capítulo, descreve-se a implementação da arquitetura dos módulos proposta, assim como o funcionamento da plataforma que os agrega. No quinto capítulo, descreve-se a metodologia utilizada para avaliar as técnicas utilizadas e os resultados obtidos. No último capítulo apresentam-se as conclusões e possíveis direções para trabalho futuro.

2. Trabalho Relacionado

Pretende-se com este capítulo descrever os tipos de técnicas, bibliotecas e ferramentas mais comuns [9] [10] em cada etapa do *workflow,* apresentado anteriormente na Figura 1.2, que descreve o fluxo de execução suportado pela plataforma desenvolvida.

O capítulo encontra-se dividido em quatro secções. A secção 2.1 descreve os dados utilizados para efeitos de teste da plataforma. Na secção 2.2 encontram-se descritas possíveis abordagens para o tratamento de um conjunto de dados, nomeadamente os *tweets* e seu conteúdo. A secção 2.3 contém uma descrição de técnicas para os tipos de aprendizagem utilizados neste projeto, assim como uma descrição das bibliotecas ou ferramentas que as suportam. Por fim a secção 2.4 possui a descrição de uma solução existente atualmente para tratar uma temática idêntica à proposta neste projeto.

2.1. Obtenção de dados

No contexto deste projeto, definiu-se que a fonte de dados a utilizar é a rede social Twitter. O Twitter é uma rede social onde os utilizadores podem comunicar através de pequenas mensagens de texto chamadas *tweets*. Estes *tweets* podem ser públicos para que todos os utilizadores da rede os possam visualizar, ou privados onde só os utilizadores que "seguem" o utilizador que fez a publicação do *tweet* podem ver. O Twitter possui um mecanismo de *retweet*, que permite que um utilizador "partilhe" o *tweet* feito por outro utilizador com os seus seguidores. Este mecanismo pode ser usado por diversos motivos, como por exemplo quando se compartilha da opinião do utilizador que publicou o *tweet* original ou se quer compartilhar essa informação para iniciar um diálogo. Pode-se descrever o Twitter como *microblogging* no sentido em que os *tweets* são breves, de 280 caracteres, mas servem para o utilizador exprimir livremente as suas opiniões ou curtas mensagens que sejam interessantes para algum outro utilizador.

No contexto deste projeto a informação a ser utilizada são os próprios *tweets* e particularmente os *tweets* criados através do mecanismo de *retweet. Retweets* estes que permitem relacionar os utilizadores intervenientes.

Na Figura 2.1 encontra-se o exemplo de um *retweet* na plataforma Twitter e na Listagem 2.1 encontra-se o objeto *JavaScript Object Notation* (*JSON)*, simplificado, que representa o *retweet* obtido através de uma chamada à API do Twitter.

Figura 2.1: *Tweet* na plataforma Twitter.

Listagem 2.1: *JSON* simplificado de um *tweet* obtido através da API do Twitter.

```
{
    "created_at": "Wed Jul 15 22:25:50 +0000 2020",
    "id": 1283528227814858754,
    "id_str": "1283528227814858754",
    "text": "RT @TwitterSupport: You may be unable to Tweet or reset your password while we review and address this incident.",
    "entities": {
        "hashtags": [],
        "symbols": [],
        "user_mentions": [
            {
                "screen_name": "TwitterSupport",
                "name": "Twitter Support",
                "id": 17874544,
                "id_str": "17874544",
                "indices": [ 3, 18 ]
            }
        ]
    },
    "source": "<a href=\"https://mobile.twitter.com\" rel=\"nofollow\">Twitter Web App</a>",
    "user": {
        "id": 783214,
        "id_str": "783214",
        "name": "Twitter",
        "screen_name": "Twitter",
        "location": "everywhere",
        "description": "",
        "url": "https://t.co/TAXQpspyHn"
    },
    "retweeted_status": {
        "created_at": "Wed Jul 15 22:18:35 +0000 2020",
        "id": 1283526400146837511,
        "id_str": "1283526400146837511",
        "text": "You may be unable to Tweet or reset your password while we review and address this incident.",
        "source": "<a href=\"https://mobile.twitter.com\">Twitter Web App</a>",
        "user": {
            "id": 17874544,
            "id_str": "17874544",
            "name": "Twitter Support",
            "screen_name": "TwitterSupport",
            "location": "Twitter HQ",
            "description": "",
            "url": "https://t.co/heEvRrl4yN"
        }
    },
    "retweet_count": 9793,
    "favorite_count": 0,
    "lang": "en"
}
```

2.2. Tratamento dos dados

Sendo os *tweets* os dados centrais do projeto, depreende-se agora a questão de como os tratar para tirar partido da informação que estes possuem. Dado que se pretende que a plataforma analise o conteúdo do *tweet* e sendo este texto produzido por um utilizador, é necessário fazer o tratamento deste tipo de dados. Este tipo de dados tipicamente são *strings*, pretende-se então processá-los de tal forma que possam ser usados como *input* nos outros módulos deste projeto, tais como a classificação e deteção de comunidades.

O texto dos *tweets* precisa de ser tratado de tal forma que se rejeite informações do seu conteúdo que pela sua frequência não possua relevância para a análise. Por exemplo, quando estamos a analisar o conteúdo de um *tweet*, de forma a categorizá-lo não é relevante palavras como "um", "uma", "de", "que", pois não acrescentam informação útil à mensagem que o utilizador tenta passar. Estas palavras podem denominar-se por *stopwords* [11] e existem bibliotecas que contêm mecanismos para a sua remoção.

Tendo um texto apenas com os dados relevantes, é necessário realizar outro passo antes de poder utilizar estes dados em algoritmos de aprendizagem automática. Visto que os algoritmos utilizados neste projeto para a criação do modelo de classificação não aceitam a representação por texto, é necessário converter as *strings* para uma representação numérica. Para tal, pode-se utilizar diversas abordagens, uma delas sendo a *bag of words* [12], na qual a ordem pela qual os termos ocorrem não é considerada. Esta abordagem consiste em particionar o documento em termos, tipicamente divide-se por espaços em branco e pontuação. De seguida, cria-se um dicionário com todos os termos presentes no universo de termos do problema e enumerando-os, tipicamente por ordem alfabética.

Na Listagem 2.2 é apresentado um exemplo simples. Em (a) estão representadas as frases que se pretende representar numericamente. Imaginando que o universo deste problema é constituído apenas por estas duas frases deve-se então criar o dicionário de palavras possíveis, enumerando-as de 0 a 13 (pois existem 14 palavras) por ordem alfabética. Como é possível verificar em (b), com o dicionário criado. Em (c) é utilizado o dicionário de (b) pois a numeração das palavras corresponde aos índices em cada *array* que representa uma frase. Cada *array* tem em cada índice o valor (0 ou 1) conforme a existência dessa palavra na frase.

Listagem 2.2: Exemplo da abordagem *bag of words* [13]

```
bags_words = ["The fool doth think he is
wise,",
"but the wise man knows himself to be a fool"]
```

(a) Documentos a converter

```
Vocabulary content:
{'a':0, 'be': 1, 'but': 2, 'doth': 3, 'fool': 4, 'he': 5, 'himself': 6,
'is': 7, 'knows': 8, 'man': 9, the': 10, think': 11, 'to': 12,
'wise': 13}
```

(b) Dicionário criado

```
[[0 0 0 1 1 1 0 1 0 0 1 1 0 1]
[1 1 1 0 1 0 1 0 1 1 1 0 1 1]]
```

(c) Resultado da conversão

Adicionalmente no tratamento de informação podem ser utilizadas técnicas que estabeleçam uma ligação entre um dado termo e a sua frequência de ocorrência. Pois assume-se que quanto maior for a frequência de um determinado termo no documento maior peso este termo deve ter. No entanto, um termo que tenha uma maior ocorrência não significa necessariamente que seja mais relevante para o contexto do documento. Para tal reduz-se o peso que o termo deve ter proporcionalmente à ocorrência do mesmo na coleção de todos os documentos. As técnicas descritas neste parágrafo são utilizadas para contextos semelhantes ao deste projeto, sendo estas *Term-Frequency* [12] e *Inverse Term-Frequency* [12].

Actualmente, existem bibliotecas como, por exemplo, a biblioteca *pandas* [14] que é *open-source* para a linguagem de programação Python [15], que disponibiliza estruturas de dados e um conjunto de ferramentas para a análise de dados, nomeadamente a conversão de dados.

Na escolha da biblioteca/ferramenta a utilizar para a transformação dos dados, pode também ter influência a escolha feita de que biblioteca/ferramenta é utilizada para realizar a classificação e *clustering*. Dado que tipicamente estas já possuem suporte para a conversão de dados e desta forma os passos a serem executados podem ser realizados na mesma biblioteca/ferramenta.

2.3. Aprendizagem Automática

Uma das técnicas muito utilizada nos dias de hoje é a aprendizagem automática, capaz de transformar informação em conhecimento. Devido à vasta quantidade de dados existente atualmente, seria um processo moroso caso se pretenda classificar os dados de forma manual. É possível facilitar a forma como esta quantidade de dados é utilizada através da utilização de aprendizagem automática. Esta técnica possibilita encontrar padrões em dados anteriormente considerados complexos, de forma automática e com pouca intervenção humana em todas as fases do processo. Esta pode assumir diferentes tipos, como por exemplo aprendizagem supervisionada, não supervisionada ou aprendizagem por reforço [16].

Embora possa assumir diferentes tipos, todas as técnicas têm componentes em comum. Existe um *dataset*, conjunto de dados, que contém características, que correspondem a elementos nos dados, que podem ser relevantes para obter conhecimento sobre os mesmos. Com o *dataset* e respetivas características identificadas é possível construir um modelo que constitui uma representação do que se pretende tratar com o algoritmo de aprendizagem automática.

Nas secções 2.3.1 e 2.3.2 vão ser descritos os tipos de aprendizagem automática, supervisionada e não supervisionada. Que correspondem a técnicas utilizadas para classificação no caso da aprendizagem supervisionada e a técnicas de *clustering* no caso

da aprendizagem não supervisionada. Não irá ser incluído o tipo de aprendizagem por reforço devido a não ser considerado no contexto deste projeto.

2.3.1. Aprendizagem Supervisionada

O objetivo da aprendizagem supervisionada é aprender como se relaciona o *input* (dados de entrada) com o *output* (dados de saída), sendo que se entende por *input* o conjunto dos dados por categorizar e *output* como estes se encontram etiquetados (*labelled)*.

Neste tipo de aprendizagem é fornecido um conjunto de dados que já se encontram etiquetados, e utilizando um algoritmo de aprendizagem é possível aprender padrões existentes nos dados. Mais tarde quando forem fornecidos novos dados, este irá aplicar os padrões aprendidos podendo assim prever as suas categorias (*labels*). Por *labelled* entenda-se que cada padrão de dados se encontra etiquetado, com uma *label* significativa que pretende dar mais informação relativamente aos mesmos.

Caso o modelo seja treinado com um pequeno conjunto de dados ou por um conjunto de dados que sejam sempre classificados com a mesma *label*, este poderá ficar "*overfitted*" o que significa que não irá ter um comportamento desejável quando for utilizado para classificar dados distintos dos quais foi treinado. É importante que o modelo seja treinado com um vasto conjunto de dados diversificados, conseguindo assim que este não fique tendencioso a regras aprendidas que apenas representam uma percentagem do universo de possibilidades com que os dados podem ser classificados. Existe então importância de que o modelo tenha capacidade de se adaptar a novos dados e consiga fazer predições adequadas, conseguindo assim uma generalização do que aprendeu.

O resultado de um modelo de aprendizagem supervisionada poderá ser categórico, tendo um conjunto finito de valores que poderá tomar, sendo assim conhecido como classificação, ou poderá ser um valor real e neste caso o resultado é conhecido como regressão.

A ***Classificação*** é uma técnica de aprendizagem supervisionada em que se pretende categorizar os dados com uma dada classe. O objetivo da aprendizagem automática nestes casos é encontrar regras que criem uma divisão clara de como os diferentes pontos de dados são separados tendo assim as diferentes regiões de classificação.

A ***Regressão*** é uma técnica também de aprendizagem supervisionada em que a ideia é prever um valor contínuo. Daí a regressão ser útil quando os problemas que se pretende resolver passam por prever um valor numérico, como por exemplo a temperatura, dadas as condições atmosféricas.

Após uma pesquisa dos métodos mais utilizados para as abordagens aqui referidas e dada a sua performance e nível de complexidade [17], alguns dos métodos que se decidiu explorar são: *i) Naive Bayes*; *ii)* árvores de decisão; *iii) Random forest:* e *iv)* Neural networks. Segue-se uma breve descrição de como os mesmos funcionam.

Neste contexto, será utilizado um exemplo, tendo em conta várias condições atmosféricas, se é possível ir jogar ténis ou não. Como características, considere-se as seguintes: *Temperature*, *Humidity* e *Wind*. As características poderão ter os seguintes valores: *hot*, *mild*, *cold* no caso de *Temperature*; *high* e *normal* no caso de *Humidity* e *weak*, *strong no caso de Wind*. Considere-se os seguintes dados de treino, da Tabela 2.1. Imaginando que existiam as seguintes condições atmosféricas de *Outlook*, *Temperature*, *Humidity*, *Wind* respetivamente com os seguintes valores *sunny*, *cool*, *high* e *strong* e pretendia-se saber se com estas condições atmosféricas seria possível jogar ténis.

Tabela 2.1: Conjunto de dados de treino do exemplo Play Tennis [18].

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

De seguida, no contexto de cada método, dá-se continuação ao exemplo descrito.

2.3.1.1. *Naive Bayes*

O método *Naive Bayes* baseia-se no teorema de *Bayes*, assumindo independência entre as características. Ou seja, a presença de uma característica específica não tem qualquer relação com a presença de outra característica nos dados. Mesmo que estas características no contexto do problema tenham uma dependência intrínseca entre elas, não é tido em consideração para o cálculo das probabilidades deste teorema. Portanto sendo *h* a hipótese de ter uma dada classe e *d* a instância de dados, tem-se o teorema de *Bayes* dado pela seguinte expressão:

$$P(h|d) = (P(d|h) * P(h))/P(d)\ . \quad (1)$$

A equação P(*h*|*d*) traduz a probabilidade daquela instância de dados, ou seja, o conjunto de características serem classificadas pela classe *h*, em função da probabilidade de existir a instância *d*. Aplicando assim o teorema a um número de hipóteses (classes) para um dado conjunto de características que representam uma instância de dados *d*, seleciona-se a hipótese (classe) que possui a maior probabilidade. Neste algoritmo, a aprendizagem de

um modelo a partir dos dados de treino é consideravelmente rápida visto que para a construção do modelo apenas é necessário o cálculo de probabilidades sem coeficientes nem operações mais morosas.

Além de permitir o seu uso para categorias, classes binárias e *multiclass* também é possível estender de forma a suportar valores reais. Para tal pode-se assumir uma distribuição gaussiana. Tipicamente utiliza-se esta distribuição pela sua facilidade de ser usada, uma vez que apenas exige o cálculo da média e do desvio padrão. No entanto, para que uma distribuição gaussiana possa ser aplicada poderá ser necessário remover *outliers*, pontos dos dados significativamente diferentes das outras observações. Ao usar este algoritmo, quando se é deparado com a existência de novos dados que representam por exemplo a existência de uma nova classe é apenas necessário atualizar as probabilidades do modelo.
Realizando o exemplo para os dados da Tabela 2.1, temos que:

$$P(d|h) * P(h)\ , \tag{2}$$

$$P(Outlook = sunny|Play = yes) * P(Play = yes) = \frac{2}{9} * \frac{9}{14}\ , \tag{3}$$

$$P(Temperature = cool|Play = yes) * P(Play = yes) = \frac{3}{9} * \frac{9}{14}\ , \tag{4}$$

$$P(Humidity = high|Play = yes) * P(Play = yes) = \frac{3}{9} * \frac{9}{14}\ , \tag{5}$$

$$P(Windy = strong|Play = yes) * P(Play = yes) = \frac{3}{9} * \frac{9}{14}\ , \tag{6}$$

$$P(d|Play = yes) * P(Play = yes)\ , \tag{7}$$

$$\Leftrightarrow \frac{2}{9} * \frac{3}{9} * \frac{3}{9} * \frac{9}{14} = 0{,}0053\ , \tag{8}$$

$$P(d|Play = no) * P(Play = no)\ , \tag{9}$$

$$\Leftrightarrow \frac{3}{5} * \frac{1}{5} * \frac{4}{5} * \frac{3}{5} * \frac{5}{14} = 0{,}0206\ , \tag{10}$$

$$P(d) = P(Outlook = sunny) * P(Temperature = cool) * P(Humidity = high) * P(Windy = strong)\ , \tag{11}$$

$$P(d) = \frac{5}{14} * \frac{4}{14} * \frac{7}{14} * \frac{6}{14} = 0{,}02186\ , \tag{12}$$

$$P(Play = yes|d) = \frac{0{,}0053}{0{,}02186} = 0{,}2424\ , \tag{13}$$

$$P(Play = no|d) = \frac{0{,}0206}{0{,}02186} = 0{,}9421\ . \tag{14}$$

Como $P(Play = no|d) > P(Play = yes|d)$, com estas condições atmosféricas opta-se por classificar como não ir jogar.

2.3.1.2. Árvore de Decisão

O método Árvore de Decisão pode ser usado tanto para classificação como para regressão. O método baseia-se numa árvore que começando na sua raiz e respondendo a umas "questões" se chega às folhas da árvore com uma "resposta" para um determinado problema. Para a construção desta árvore é utilizado um conjunto de dados etiquetado em

que é importante saber que características se devem considerar, em que condições se deve realizar cada partição, e quando se deve parar de fazer partições dos dados.

Tipicamente para determinar quais as partições a realizar no conjunto de dados é utilizado o método *Recursive Binary Splitting* [7], que tendo em consideração todas as características, realiza diferentes partições que possuem um custo associado. Idealmente, pretende-se ter o menor custo possível, optando-se assim pela partição que dá origem a um menor custo.

O mecanismo de partições que é utilizado na criação da árvore de decisão pode dar origem a uma árvore que se encontra *overfitted*, ou seja, encontra-se demasiado dependente dos dados de treino. Como tal, de forma a tentar evitar que isso ocorra, pode-se definir um número mínimo de dados de treino que são usados como *input* em cada folha, evitando assim que sejam gerados ramos da árvore que sejam *overfitted* a um pequeno conjunto de dados. Além disto, pode-se também definir um número máximo de profundidade que se deseja, evitando assim o crescimento excessivo da árvore.

Um algoritmo também muito usado na construção de árvores de decisão é o *Induction Tree* (ID3) [19]. Este algoritmo é iterativo e o seu objetivo é construir uma árvore de decisão a partir de um conjunto de dados. Recorre a duas medidas: entropia e ganho de informação. Em cada iteração seleciona-se o atributo, que tenha um maior ganho de informação. Para exemplificar como o algoritmo funciona e utiliza estas medidas, realizou-se os cálculos de forma a construir a árvore de decisão que teve por base o conjunto de dados apresentados na Tabela 2.1.

Seguindo com o conjunto de dados utilizado no método anterior, temos os seguintes valores de entropia.

Entropia: $$H(x) = -p * log2(p) - p\_ * log2(p\_)\ .$$ (15)

Entropia do conjunto de dados: $H(x) = -\frac{9}{14} * log2\left(\frac{9}{14}\right) - \frac{5}{14} * log2\left(\frac{5}{14}\right) = 0{,}94\ .$ (16)

Deste modo, o cálculo do ganho de informação para uma característica, e os respetivos valores são:

$$Gain(S, wind) = H(x) - \frac{8}{14} * H(Strue) - \frac{6}{14} * H(Sfalse),$$ (17)

$$Gain(S, Wind) = 0{,}048,$$ (18)

$$Gain(S, Outlook) = 0{,}246,$$ (19)

$$Gain(S, Humidity) = 0{,}151,$$ (20)

$$Gain(S, Temperature) = 0{,}029.$$ (21)

Desta forma opta-se pela característica que possui o maior ganho de informação. De seguida realiza-se o cálculo do ganho tendo em conta o valor da característica no ramo em que estamos. Na Figura 2.2 encontram-se os passos para a criação da árvore de decisão

para o exemplo descrito anteriormente na Tabela 2.1, em que os valores entre parêntesis retos correspondem às entradas do conjunto de dados que pertencem a cada classe, em que + é quando Play=yes e – é quando Play=no. Quando um desses valores se encontra a 0, quer dizer que já se sabe a predição daquele caminho na árvore.

Para as mesmas condições atmosféricas do exemplo anterior tem-se que Play=no, classificação obtida com as características também descritas anteriormente (a-c).

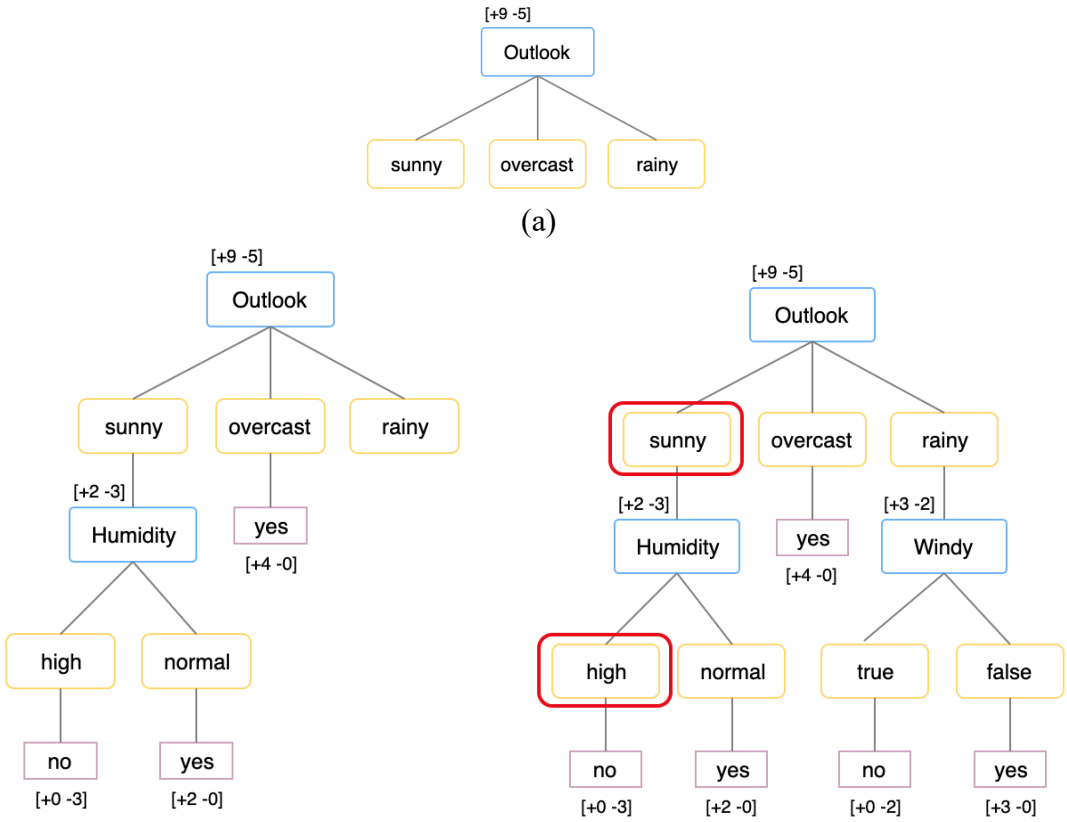


Figura 2.2: Passos de criação de árvore de decisão.

2.3.1.3. Random Forest

O método *Random Forest* consiste num conjunto de modelos de árvores de decisão. Possui dois conceitos chave, sendo estes *booststrapping* e *bagging* [20]. *Bootstrapping* consiste numa técnica de amostragem em que é retirada aleatoriamente uma amostra do conjunto de dados disponíveis. *Bagging* consiste na agregação de árvores de decisão, criando assim um total de *x* árvores de decisão que são usadas no treino do modelo para *y* conjuntos de dados de treino retirados aleatoriamente através da técnica de *bootstrapping*.

O resultado com a predição, é a decisão que tiver mais votos, ou seja, a que mais árvores tiver previsto do conjunto das árvores que constituem o modelo.

Este método possui vantagens perante uma única árvore de decisão, pois esta possui uma alta variância tendo tendência a ficar *overfitted* aos dados que foram usados para criar o

modelo. Usando a técnica de *bagging*, retira-se esta tendência uma vez que o valor de *output* passa a ser a predição que teve mais votos, de um conjunto de árvores de decisão criadas por subconjuntos dos dados aleatoriamente. Além disto, descorrelaciona as árvores dando uso à técnica de *bootstrapping*, uma vez que introduz partições em conjuntos aleatórios de características. Em termos de tempos, o tempo de aprendizagem do modelo é mais rápido do que nas árvores de decisão uma vez que o foco está apenas num subconjunto das características.

2.3.1.4. Redes neuronais

As redes neuronais tiveram inspiração em parte da observação do sistema biológico de aprendizagem, composto por redes complexas de neurónios interconectados.

Assim, uma rede neuronal consiste em transformar um conjunto de dados de *input* num *output* através de um conjunto de cálculos. Consiste num conjunto de unidades denominadas por neurónios, tipicamente em colunas que constituem uma camada da rede neuronal. A organização de uma rede neuronal, como ilustrada na Figura 2.3, é constituída por diversas camadas, sendo a primeira camada a que recebe os dados de entrada, as camadas intermédias são denominadas *hidden* (escondidas) e por uma última camada que contém os possíveis resultados de saída.

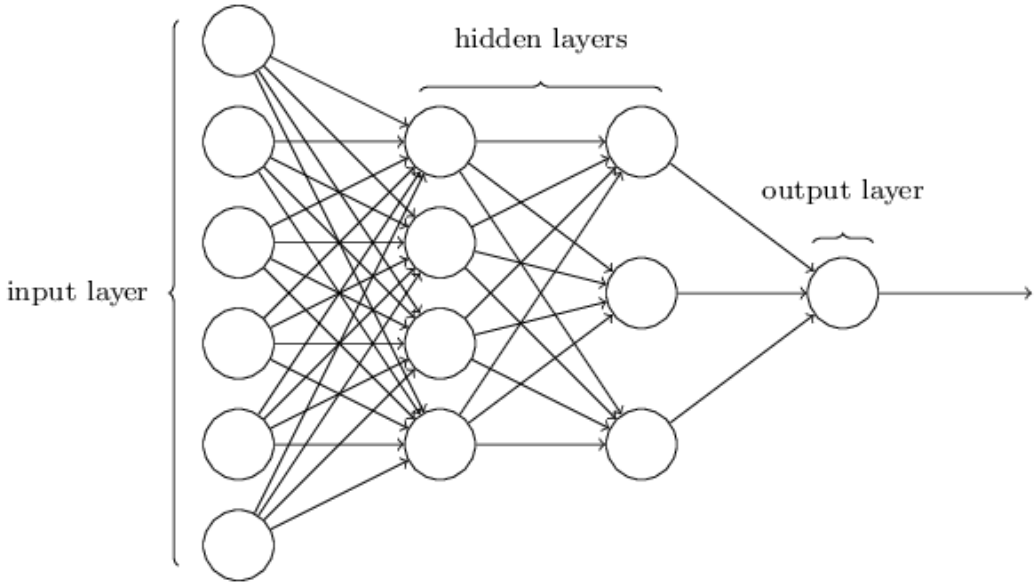


Figura 2.3: Exemplo de rede Neuronal, designação das suas camadas [21].

Um neurónio desempenha um conjunto de operações que permitem através dos dados de entrada recebidos produzir um resultado, como ilustrado na Figura 2.4.

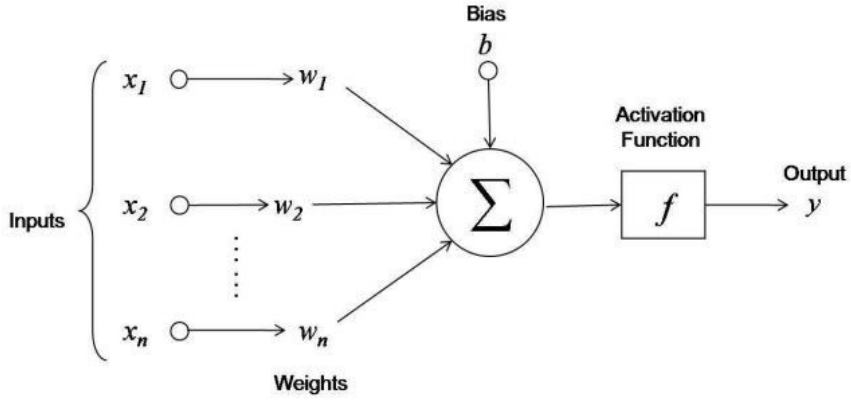


Figura 2.4: Operações de um neurónio [22].

Representado por x_n entenda-se o que o neurónio recebe, e por w_n entenda-se o conjunto de pesos que determinam o peso da conexão entre um dado neurónio de uma dada camada e um neurónio da camada anterior que se conecta com ele.

É necessário ajustar os pesos, para que com os *inputs*, sendo estes vários exemplos de treino, a rede produza o mesmo *output* que o *output* esperado para aquele exemplo de treino. De forma a determinar o vetor de pesos ideais para o exemplo de treino tipicamente é utilizado o Gradiente descendente [18], numa tentativa de minimizar o erro entre o *output* da rede e o *output* desejado. O valor *b*, tal como os pesos é utilizado para ajustar o *output* aos exemplos de treino, sendo um valor constante que é adicionado à equação linear do *input* com os pesos. A função de ativação é utilizada de forma a definir o *output* através do valor calculado anteriormente com o conjunto de *inputs*. Existem vários tipos de função de ativação sendo as mais comuns *Binary Step Function*, *Linear Function* e *Sigmoid* [23].

Para a rede neuronal aprender, tipicamente é utilizada uma técnica de *Back-propagation* [18], ilustrada na Figura 2.5 que através do erro de *output*, o resultado não corresponder ao esperado, permite assim alterar os pesos de toda a rede neuronal de forma a alterar o resultado produzido.

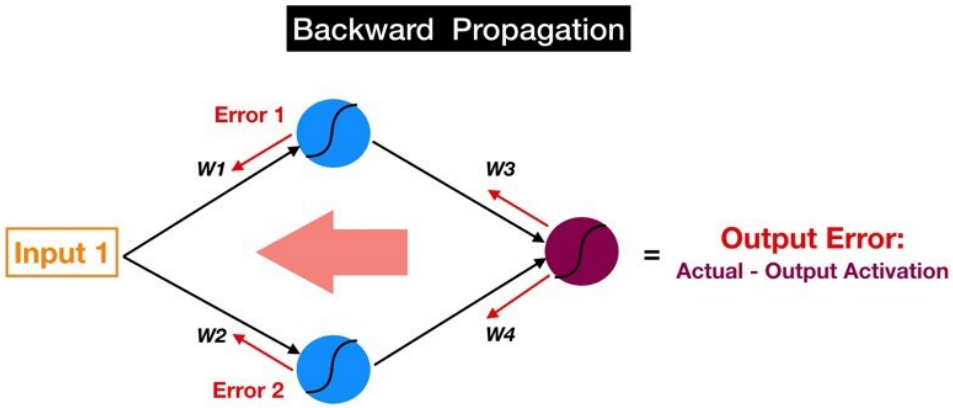


Figura 2.5: *Back-propagation* numa rede neuronal [23].

2.3.2. Aprendizagem não supervisionada

Na aprendizagem não supervisionada apenas são fornecidos os dados de entrada sem que estes se encontrem *labelled*. Com a remoção da componente supervisionada, este tipo de aprendizagem pode tornar-se mais complexo pois isto significa que o problema se encontra menos definido.

Uma forma de aprendizagem não supervisionada é agrupamento de dados, que consiste em criar grupos de dados, os quais se assemelham entre si, mas que se distinguem de alguma forma de dados de outros grupos.

Existem outras formas de aprendizagem não supervisionada, sendo por exemplo por uma forma de associação, onde se pretende encontrar regras que descrevam os dados. Técnicas que envolvam associação são mais utilizadas, por exemplo, para criar recomendações para clientes com base no que estes já compraram.

Neste trabalho, vão ser abordadas as seguintes técnicas: *i) k-Means; ii) Gaussian Mixture;* e *iii) Louvain*.

2.3.2.1. k-Means

O objetivo do método k-Means passa por agrupar pontos de dados semelhantes. O *k* no nome deste método vem exatamente do número de grupos que se pretende encontrar no conjunto de dados, sendo este um parâmetro do algoritmo que implementa o método.

Este método agrupa os dados em *k* grupos diferentes, em que cada grupo possui um centroide, que se trata de uma representação imaginária/real do centro do *cluster*. Após obter os centros dos *clusters* localizados aleatoriamente, cada ponto dos dados é alocado e passa a pertencer ao *cluster* que tem o seu centroide mais próximo. Para saber o *cluster* a que cada ponto de dados pertence, é mais frequentemente considerada a distância Euclidiana [24] a todos os centroides e opta-se pelo que se encontra a menor distância. Após a alocação de cada ponto dos dados, é realizada a média da localização de todos os pontos de dados pertencentes a cada centroide e dessa forma atualiza-se a localização do centroide do respetivo *cluster*. Este processo termina quando os *clusters* se encontram completos, ou seja, todos os pontos de dados já se encontram estabelecidos ou o número de iterações termina.

Na Figura 2.6 encontra-se um exemplo iterativo do processo, onde inicialmente os pontos verdes são os pontos dos dados e a cruz (x) azul e vermelho são os centroides. À medida que se atribui os pontos ao respetivo centroide, no fim de cada iteração atualiza-se também a posição do centroide para ficar centrado nos pontos de dados que lhe pertencem. O exemplo terminou, quando calculadas as distâncias não houve nenhum ponto dos dados que causasse uma mudança no *cluster* onde se encontrava.

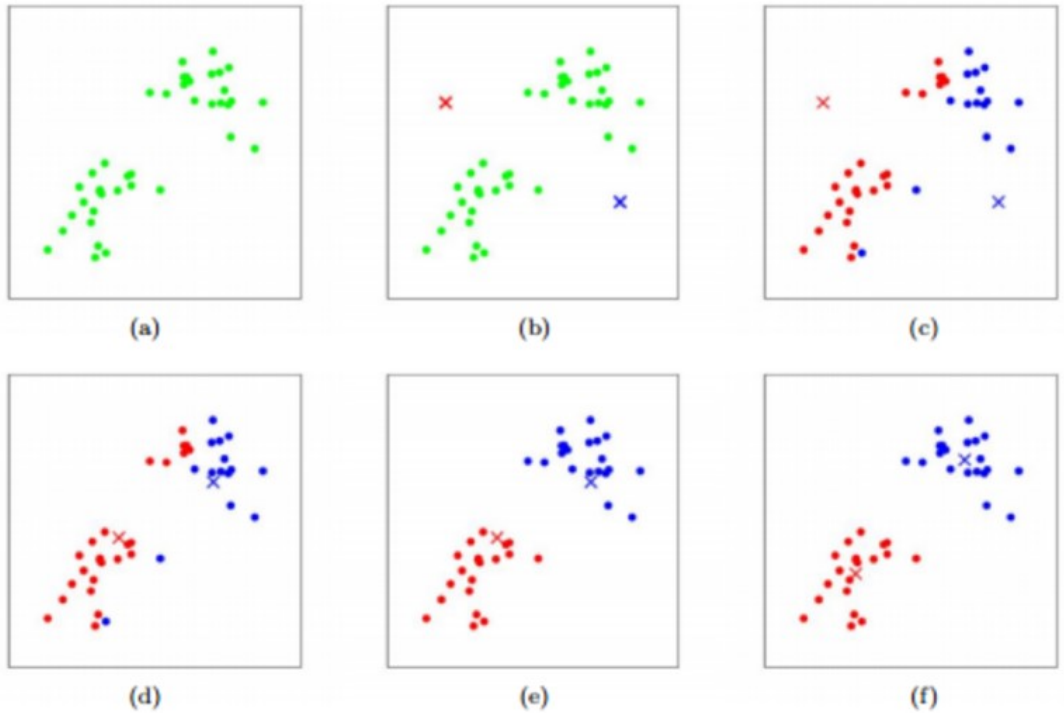


Figura 2.6: Passos do algoritmo k-Means (a-f) [8].

2.3.2.2. Gaussian Mixture

O método Gaussian Mixture é semelhante ao *k-Means* anteriormente referido, diferindo, pois, consegue descobrir *clusters* que não sejam necessariamente circulares. Dado o *k-Means* utilizar a distância Euclidiana no cálculo dos *clusters*, este apenas vai encontrar *clusters* circulares não tendo em consideração a direção em que o maior número de pontos dos dados se encontra.

Neste método, para cada *cluster* existe uma distribuição Gaussiana [25] que é definida por uma média que corresponde ao centro da distribuição e variância que representa o quão "espalhados" os dados se encontram.

O método inicia com centro de *clusters* aleatórios, e vai associar cada ponto dos dados a um dos *clusters*. Para isso vai calcular a probabilidade de cada um dos pontos pertencer a cada um dos *clusters* criados inicialmente. Por fim reajusta as médias e as variâncias de forma a criar novos *clusters* de acordo com os pontos de dados que lhe pertencem. Estes passos são repetidos até que não haja alterações nos *clusters*. Utilizando a variância assume-se que um *cluster* com uma distribuição Gaussiana com maior variância possui maior probabilidade de um ponto pertencer a esse *cluster*.

A Figura 2.7 ilustra visualmente os passos do método referido, verificando os *clusters* com diferentes formas, e após 20 iterações termina pois não existem mais alterações nos *clusters* criados.

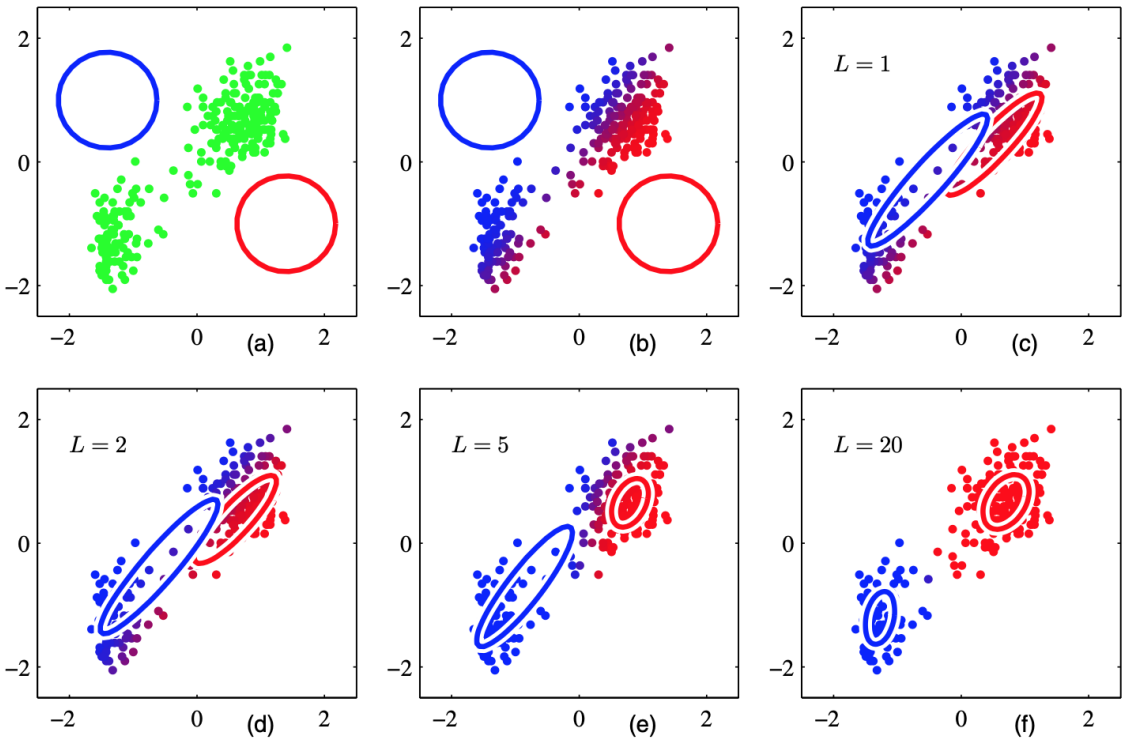


Figura 2.7: Passos do algoritmo *Gaussian Mixture Model* (a-f) [26].

2.3.2.3. Louvain

O objetivo do método Louvain é a descoberta de comunidades numa rede de conexões. Comunidades são grupos de nós dentro de uma rede, que se encontram conectados de uma forma mais densa do que outros nós que não pertencem à mesma comunidade. Como forma de definir as comunidades utiliza-se uma métrica chamada modularidade que consiste em quantificar a qualidade de agrupar um dado nó a uma dada comunidade. Assim compara-se quão mais densamente o nó se encontra conectado com a comunidade que foi atribuído do que em média se fosse atribuído a uma comunidade aleatoriamente.

O método consiste na aplicação de dois passos em várias passagens. O primeiro passo, tem uma abordagem "*greedy*", na medida em que pretende maximizar a modularidade das partições encontradas na rede. Desta forma, inicialmente, atribui-se uma comunidade a cada nó. O segundo passo consiste em calcular a modularidade entre os nós vizinhos caso fosse para estes formarem uma nova comunidade. Em caso de um ganho positivo na modularidade então estes passam efetivamente a constituir uma nova comunidade, retirando um dos nós da sua comunidade e atribuindo-o à comunidade do seu nó vizinho que constitui um aumento da modularidade. Este passo é repetido para todos os nós até que não exista mais nenhuma alteração na atribuição dos nós às comunidades.

Desta forma, termina-se a primeira fase deste algoritmo. As próximas fases consistem em criar uma nova rede e repetir os dois passos iniciais, mas toma-se as comunidades criadas anteriormente como nós individuais, tentando-se agregar comunidades de forma a refletir um ganho na modularidade. Este processo termina quando já não existem alterações nas comunidades e foi obtido um máximo de modularidade para aquela rede.

O algoritmo é eficiente, sendo que ao longo do tempo diminui o número de comunidades e o ganho da modularidade não exige cálculos complexos. A maior percentagem do tempo de execução prende-se então na fase inicial em que existem tantas comunidades quanto o número de nós e é necessário o cálculo da modularidade para muitos casos [27].

Uma das características deste método é que um nó poderá ser considerado mais que uma vez no cálculo do ganho da modularidade e que a ordem pela qual os nós são considerados não parece ter influência no ganho da modularidade. No entanto, considerar um nó múltiplas vezes no cálculo do ganho da modularidade, reflete-se no tempo computacional pois pode dar a origem a mais mudanças nas comunidades e por consequência mais cálculos a serem executados.

A Figura 2.8 [27] pretende ilustrar os passos que o constituem. Neste algoritmo, inicialmente cada nó constitui uma comunidade, e a partir do ganho da modularidade vai-se agrupando as comunidades representadas na Figura 2.8 pelas diferentes cores.

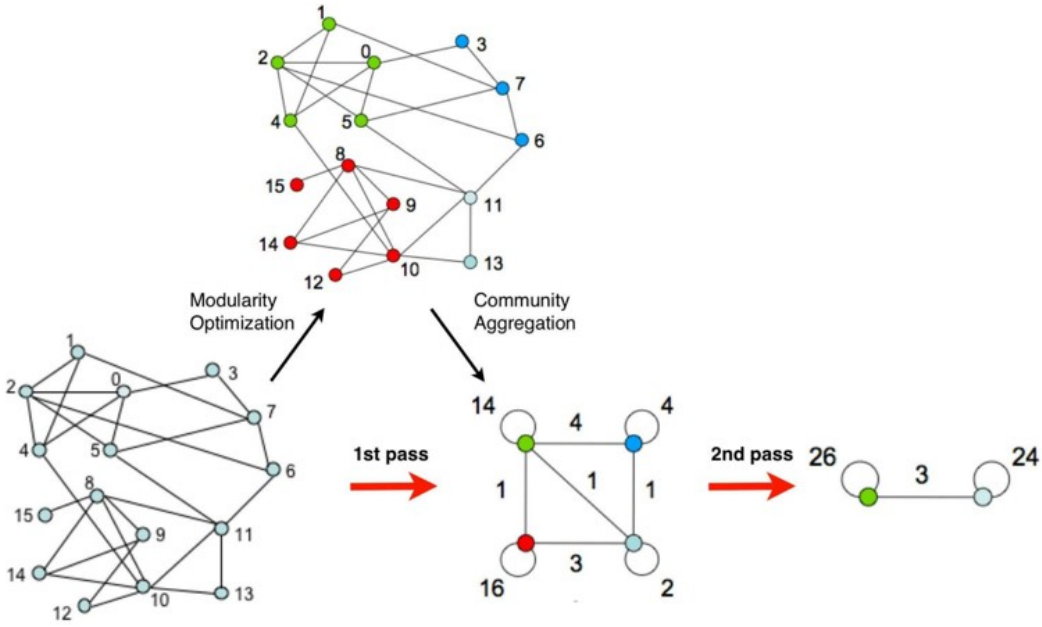


Figura 2.8: Passos do algoritmo de Louvain [27].

2.3.3. Bibliotecas e plataformas de suporte

Existem diversas bibliotecas e plataformas que possuem suporte para os dois tipos de aprendizagem automática referidos nas secções 2.3.1 e 2.3.2. Entre elas existe a biblioteca Scikit-learn [28] que é *open-source,* e é utilizada para aprendizagem automática na linguagem Python. Esta possui vários algoritmos de classificação, regressão e *clustering*, incluindo implementação dos algoritmos referidos nesta secção, nomeadamente: *i) Naive Bayes*; *ii)* Árvores de decisão; *iii) Random Forest*; *iv) Neural Network; v) k-Means; vi) Gaussian Mixture* e *vii) Louvain*.

Existe também a *framework* Apache Spark [29] que por ser utilizada no desenvolvimento deste projeto irá ser descrita mais detalhadamente. Na Figura 2.9, encontra-se o ecossistema [29] que constitui o Apache Spark. Como se pode verificar são

disponibilizadas várias bibliotecas. No contexto deste projeto para a implementação da classificação e a deteção de comunidades é principalmente relevante a biblioteca MLlib [30] de aprendizagem automática. A biblioteca MLlib possui várias implementações de algoritmos tanto para a classificação como para *clustering,* entre estes e dos algoritmos referidos ao longo da secção anterior estão incluídos os seguintes: *i) Naive Bayes*; *ii)* Árvores de Decisão; *iii) Random Forest*; *iv) Neural Network; v) k-Means;* e *vi) Gaussian Mixture*.

Apache Spark é um mecanismo de computação unificada com um conjunto de bibliotecas para processamento de dados em paralelo que irá ser realizado em *clusters*. O Spark possui também suporte para múltiplas linguagens de programação. Pode executar tanto num computador como num *cluster* de múltiplos servidores. As aplicações Spark executam como conjuntos de processos independentes num *cluster* que é coordenado pelo *SparkContext*. O Spark delega as tarefas que lhe são atribuídas através do objeto *SparkContext,* que por sua vez liga-se a um *Cluster Manager*. O *Cluster Manager* é responsável por gerir as tarefas pelos diferentes nós de execução. Um nó de execução consiste num conjunto de processos capaz de executar em paralelo para realizar as tarefas que lhe foram atribuídas. Na Figura 2.10 encontra-se representada a arquitetura de *cluster management* [31] do Apache Spark.

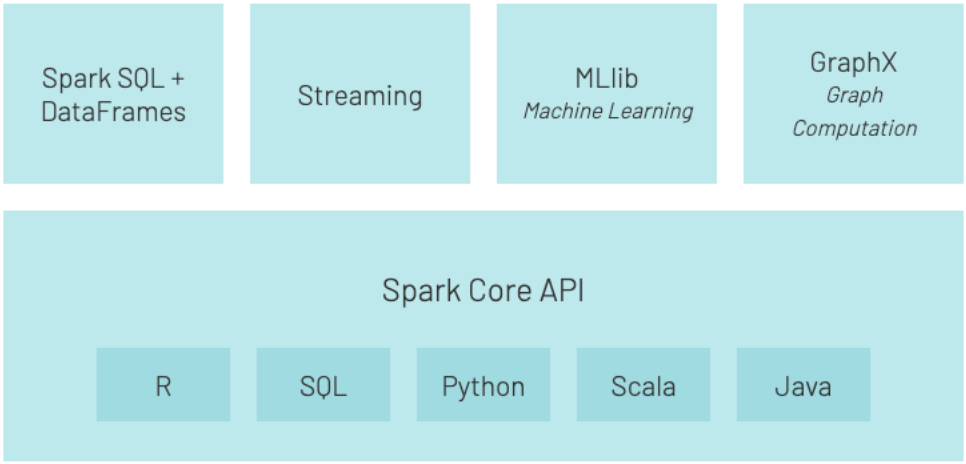


Figura 2.9: Ecossistema Apache Spark [29].

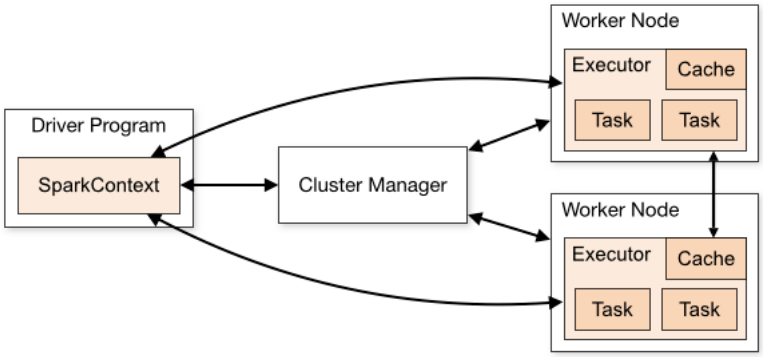


Figura 2.10: *Workflow cluster management* Apache Spark [32].

2.4. Soluções Existentes

Uma das soluções encontradas que possui mais funcionalidades em comum com a plataforma desenvolvida é a solução *MonkeyLearn* [33]. Esta solução oferece análises tanto ao nível de tópicos como uma análise sentimental sobre um texto. Possui integração com diferentes fontes de dados, dependendo do plano em utilização. Ou o utilizador pode utilizar os dados que pretender, através por exemplo de um ficheiro ou de uma base de dados. A análise sentimental é feita sem precisar de intervenção por parte do utilizador, e classifica conforme o tom, apreciativo ou negativo, que a informação apresenta. A análise de tópicos consiste na aplicação de aprendizagem supervisionada a um conjunto de dados. Como tal, a solução permite que o utilizador faculte os dados com que pretende treinar um modelo de classificação e o utilizador poderá classificá-los com os tópicos que entender.

Foram encontradas diversas ferramentas de análise sentimental, no entanto a solução referida acima foi a que possui funcionalidades mais próximas com a solução proposta neste projeto. Nomeadamente a classificação de dados, pois já trata desta temática. No entanto, a solução proposta neste projeto apresenta outras funcionalidades, pois além da classificação também disponibiliza uma forma de agregar os dados com associações específicas encontradas em dados provenientes de redes sociais, no contexto da solução proposta, o Twitter. Outro dos motivos é que a solução descrita nesta secção tem vertentes de planos pagos, fazendo que o plano não pago apenas possibilita a customização de um modelo de classificação, ficando assim o utilizador limitado. Por oposição, na solução proposta o utilizador pode criar diversos modelos de classificação.

3. Arquitetura da solução

Pretende-se com este capítulo descrever os vários módulos que constituem a plataforma *TwtAnalysis*. Cada módulo possui um conjunto de funcionalidades e objetivos cujos detalhes de implementação serão explicados no Capítulo 4.

O capítulo encontra-se dividido em quatro secções. A secção 3.1 descreve os dados utilizados na solução. Na secção 3.2 encontra-se descrito o processamento a realizar sobre os dados. A secção 3.3 descreve as etapas necessárias para a classificação dos dados tanto no treino do modelo como para realizar predições. Por fim, a secção 3.4 descreve as etapas necessárias à deteção de comunidades, mais especificamente as associações consideradas.

A Figura 3.1, apresenta o diagrama da arquitetura da plataforma desenvolvida. A plataforma desenvolvida possui uma componente de aplicação web para facilitar a interação com o utilizador e uma componente que gere os pedidos e as respostas dos módulos que suportam as funcionalidades necessárias para que exista o fluxo de processamento desejado. Como já foi referido na secção 1.1, existem dois fluxos de processamento possíveis no *workflow* da plataforma. Um deles é o treino que se encontra ilustrado na figura como (a) que após processar e filtrar os *tweets* classificados provenientes do utilizador, obtém o modelo de classificação e retorna esses dados ao utilizador. Sendo o outro a possibilidade de realizar predições, classificar novos *tweets* e realizar a deteção de comunidades, encontrando-se este ilustrado por (b). As etapas que são comuns aos dois fluxos não possuem a designação (a,b).

O fluxo da plataforma deve funcionar pela ordem estabelecida, não sendo possível realizar os passos do segundo fluxo sem que tenha existido o primeiro fluxo.

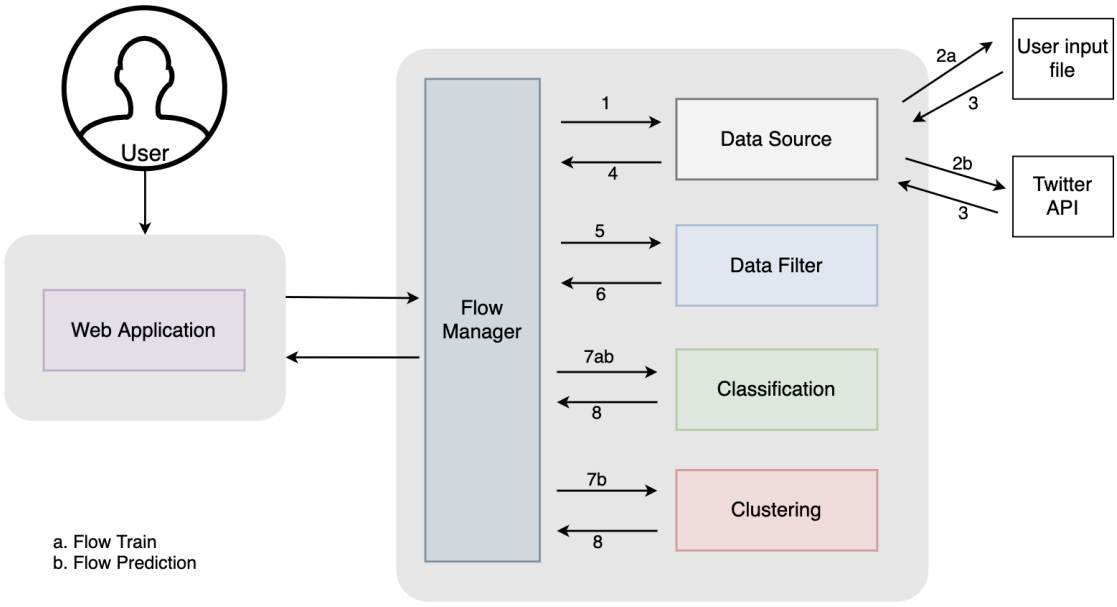


Figura 3.1: Diagrama da arquitetura da plataforma desenvolvida.

As secções seguintes detalham as funcionalidades de cada módulo necessário ao fluxo de processamento da plataforma.

3.1. Obtenção de dados

Os dados utilizados no contexto deste projeto são *tweets,* como referido na secção 1.1, no entanto estes podem ser adquiridos através de um ficheiro ou podem ser obtidos através da API do Twitter. Quando os *tweets* são obtidos através de um ficheiro, este pode conter diversa informação que não é necessária devendo assim ser filtrada.
Na obtenção dos *tweets* através da API do Twitter, como referido na secção 2.1, existem diversos campos, no entanto a informação necessária no contexto deste projeto é apenas a que se encontra na Listagem 3.1. Os campos necessários de cada *tweet* são os seguintes: *i) id*, identificador do *tweet*; *ii) text*, texto do *tweet*; *iii) user_id*, identificador do utilizador; *iv) user_screen_name*, nome do utilizador na plataforma Twitter; *v) retweeted_user_screen_name*, nome do utilizador que fez o *tweet* original; e *vi) lang*, linguagem.

Pretende-se que os *tweets* em análise se encontrem na língua que o utilizador define e que contenham pelo menos uma das palavras do conjunto de palavras-chave. A filtragem dos *tweets* pelas palavra-chave é aqui introduzida como forma de existir uma seleção de *tweets* que representem um tema ou determinado assunto de interesse. Estes parâmetros são utilizados no pedido à API do Twitter de forma a obter os *tweets* que correspondem à língua pretendida e que estes respeitem a condição de existência das palavras-chave.

Listagem 3.1: Campos relevantes na obtenção de um *tweet*.

```
{
    "id": id,
    "text": text,
    "user_id": user.id,
    "user_screen_name": user.screen_name,
    "retweeted_user_screen_name": retweeted_status.user.screen_name,
    "lang": lang
}
```

3.2. Tratamento dos *tweets*

No módulo de tratamento dos *tweets* pretende-se realizar a filtragem do seu conteúdo e remover *stopwords*. Na Figura 3.2 está presente um diagrama que pretende ilustrar o processamento realizado por este módulo.

Num *tweet* existe conteúdo não relevante, que no caso da plataforma *TwtAnalysis* não será analisado e como tal deve ser excluído, de forma a não criar "ruído" no conteúdo que realmente importa analisar. Um desses casos é URL para conteúdos externos, embora em determinados contextos esta informação seja relevante. No entanto, assumiu-se para este trabalho não considerar esta informação. Deve ainda ser realizada a remoção das *stopwords* contidas nos *tweets* pois, como referido na secção 2.2, estas não acrescentam informação relevante ao conteúdo que se pretende analisar. Assim, obtém-se um conjunto de dados preparado para que seja feita a análise proposta.

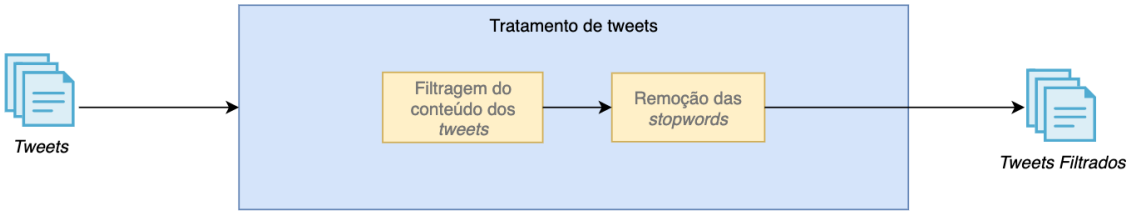


Figura 3.2: Diagrama do módulo de tratamento de *tweets*.

3.3. Classificação

Um dos requisitos da plataforma proposta é proceder à classificação categórica dos *tweets*, daí depreende-se a necessidade de existir um módulo que trate de todo este processo tendo como dados de entrada um conjunto de *tweets*. Estes são previamente filtrados pelo módulo descrito na secção anterior, o módulo descrito nesta secção produz como resultado o modelo de classificação ou *tweets* categorizados, dependendo da fase da classificação em que se encontra, treino ou predição.

O módulo pressupõe dois fluxos distintos de classificação, como visto nos casos de utilização na secção 1.1, na Figura 1.1. Num dos fluxos, o utilizador pode treinar o modelo, em que o *input* do módulo é um conjunto de dados, facultado pelo utilizador, previamente classificados. Neste caso, os dados vão ser utilizados na criação de um modelo de classificação capaz de fazer predições, tendo como *output* o próprio modelo. No outro fluxo, o utilizador pode obter predições, em que nesse caso o *input* do módulo

é o modelo de classificação e um conjunto de *tweets* filtrados por classificar e o *output* são os *tweets* classificados.

A Figura 3.3 pretende ilustrar o funcionamento do módulo, os possíveis *inputs* e *outputs*, assim como o processamento realizado, conforme as fases de classificação descritas acima.

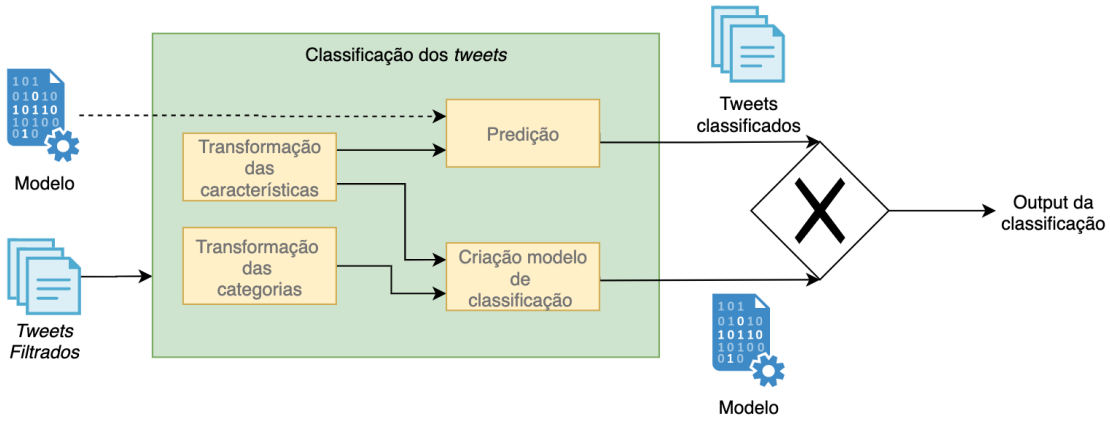


Figura 3.3: Diagrama do módulo de Classificação dos *tweets*.

No processamento do módulo, tanto para a criação do modelo de classificação como na sua utilização, para realizar predições, é necessário realizar uma transformação dos *tweets* filtrados. Tipicamente as técnicas de classificação automática não suportam como *input* dados representados em texto, e uma vez que este é o caso do conteúdo dos *tweets* a analisar é necessário proceder à sua transformação. Esta transformação deve ser realizada tanto a nível do conteúdo que se vai analisar, como das categorias, caso existentes, e no caso em que estejam também representadas em texto. Propõe-se então, para a transformação das características, a utilização da informação da frequência dos termos. Neste contexto, são as palavras que existem no conjunto dos *tweets*. Para isto utiliza-se a abordagem *Term Frequency e Inverse document frequency* [34], a qual determina a importância de uma palavra através da sua frequência no documento. Para isto tem como pressuposto que para uma palavra ser considerada relevante num determinado *tweet* deve ter uma frequência de ocorrência elevada nesse *tweet* mas ser pouco frequente noutros *tweets*.

O processo que é realizado pelo módulo descrito nesta secção é definido por um *pipeline* que se inicia pela transformação dos dados e que finda com um modelo de classificação ou com um conjunto de dados com as respetivas predições.

3.4. Deteção de Comunidades

Além da categorização também se pretende a análise de comunidades existentes com a informação inerente nos *tweets,* em particular a associação entre utilizadores traduzida pelos *tweets* que foram realizados através do mecanismo de *retweet*.

Para a obtenção de comunidades, é necessária uma filtragem do conjunto de tweets. Esta filtragem serve para se adequarem ao contexto da deteção de comunidades, sendo o resultado da filtragem utilizado para criar uma rede que os relaciona. Por fim, a rede criada é utilizada para proceder ao agrupamento dos utilizadores e desta forma obter as comunidades existentes.

A filtragem dos *tweets* recebidos como *input* é necessária pois estes contêm tanto os *tweets* como os *retweets*. Como referido anteriormente, apenas *tweets* criados através do mecanismo de *retweet* são relevantes na deteção de comunidades, devendo filtrar os restantes.

O diagrama da Figura 3.4 representa o processamento realizado por este módulo até à obtenção das comunidades.

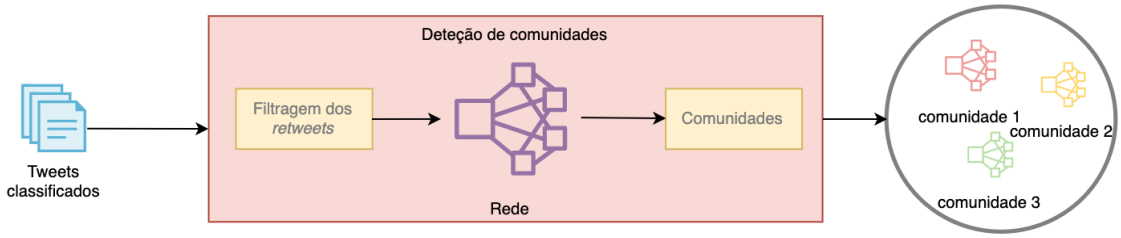


Figura 3.4: Diagrama do módulo de Deteção de comunidades.

Para a descoberta de associações entre utilizadores, é necessária a existência de uma rede que os relaciona entre si. Existem várias abordagens para criar estas associações tendo em conta a informação existente nos *tweets*. Um exemplo seria optar pela informação dos seguidores de um dado utilizador, esta abordagem seria interessante se o interesse fosse analisar o "público" de um dado utilizador. No entanto, no contexto deste projeto optou-se por utilizar a informação do mecanismo de *retweet* na medida em que traduz a influência que um utilizador possa ter sobre o outro.

A realização de um *retweet* significa que existiu uma partilha de opiniões, no caso em que o utilizador A faz muitos *retweets* ao utilizador B podemos concluir que as opiniões de B têm influência no comportamento de A. Assim, construindo uma rede utilizando os *retweets* e pegando no exemplo concreto do projeto HATE, como exemplo de utilização desta plataforma, seria interessante aferir a orientação política entre as várias comunidades encontradas para perceber de que forma estas são constituídas. Podendo esta informação ser útil para adaptar uma campanha política a um dado grupo de utilizadores.

Tendo a informação da rede, deve proceder-se ao agrupamento dos dados de tal forma, que utilizadores de determinado grupo tenham uma maior relação entre si do que com utilizadores que foram agrupados noutro grupo. Além disso, tirando partido da informação obtida no módulo de classificação, com a categorização dos *tweets* é possível determinar dentro de uma dada comunidade qual é a categoria que predomina e em que medida isso se relaciona com as categorias que predominam nos *tweets* realizados, individualmente, por cada utilizador da comunidade.

4. Implementação

Neste capítulo são referidas diversas tecnologias e técnicas por forma a cumprir os requisitos e funcionalidades descritos na secção 1.1. Para tal, a plataforma *TwtAnalysis* possui um *pipeline* que na sua composição utiliza os módulos descritos no capítulo anterior.

O capítulo encontra-se dividido em seis secções. A secção 4.1 descreve as tecnologias consideradas para a implementação da plataforma TwtAnalysis. Na secção 4.2 encontram-se detalhes de implementação para a obtenção de dados. A secção 4.3 descreve as filtragens realizadas sobre os dados. Na secção 4.4 encontram-se descritos os detalhes de implementação para a classificação de dados, quais os algoritmos que a plataforma disponibiliza para realizar a classificação e respetiva parametrização. A secção 4.5 contém uma descrição das associações utilizadas para o agrupamento dos dados, quais os algoritmos disponibilizados para esse efeito e respetiva parametrização. Por fim, a secção 4.6 apresenta o funcionamento da plataforma desenvolvida, *TwtAnalysis*.

A Figura 4.1 apresenta um esquema com a relação dos módulos da solução. A solução baseia-se na obtenção dos *tweets* através de um ficheiro facultado pelo utilizador ou utilizando a API do Twitter. Obtidos os dados, como descrito na secção 3.1, é então necessário processá-los tirando partido do módulo descrito na secção 3.3. Findo o processamento inicial dos *tweets*, o *pipeline* da solução pode tomar dois fluxos: *i)* criar um modelo de classificação; ou *ii)* utilizar um modelo de classificação para realizar predições. Para o fluxo *i)* utiliza-se o módulo descrito na secção 3.3. No fluxo *ii*), as predições incluem a categorização dos *tweets* e a deteção das comunidades existentes entre utilizadores do Twitter. Para a categorização é utilizado o módulo descrito na secção 3.3 e para a deteção de comunidades é utilizado o módulo descrito na secção 3.4.

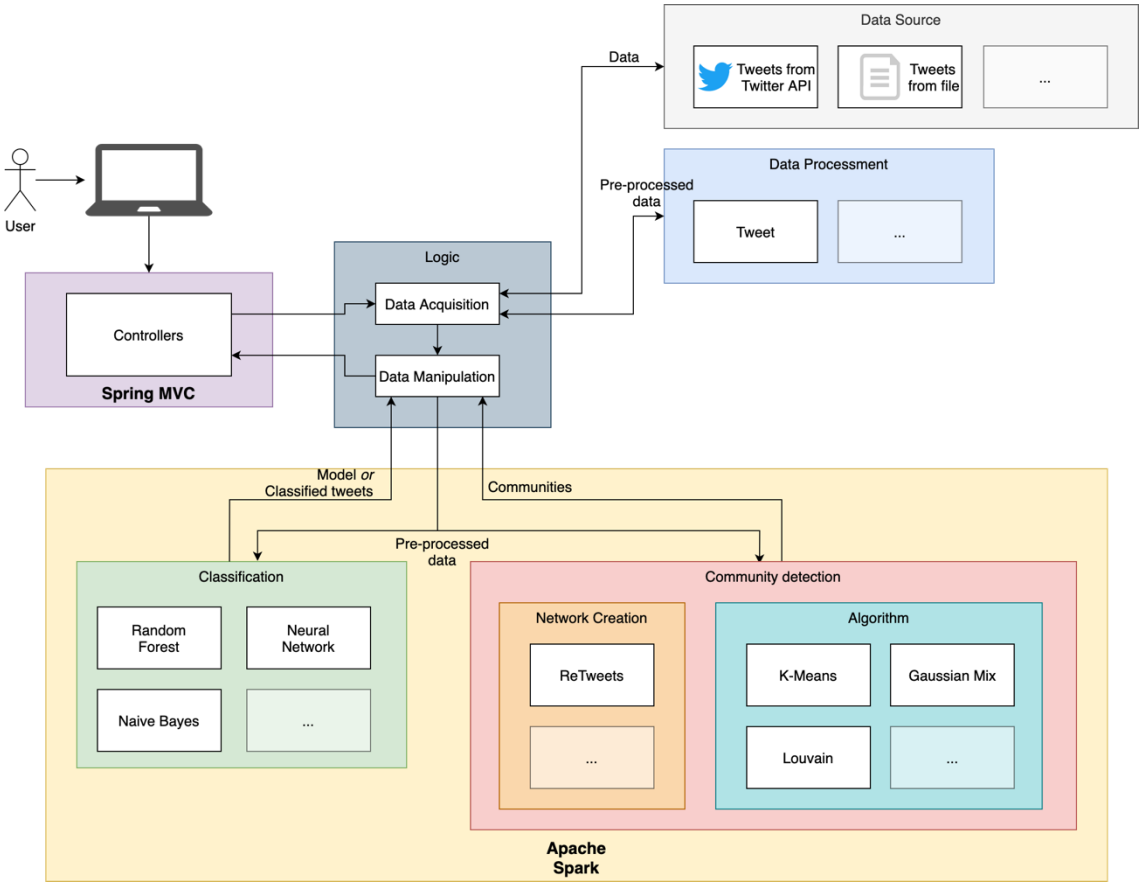


Figura 4.1: Diagrama da solução implementada.

Neste capítulo exemplifica-se, através dos dados do projeto HATE, as etapas realizadas pelos módulos existentes na plataforma *TwtAnalysis*. Como tal, imagine-se o cenário em que já existe um modelo que foi treinado para categorizar *tweets* quanto à sua orientação política, ou seja, se é de "Direita", "Esquerda" ou caso não tenha sido possível concluir "N/D". Tendo um conjunto de *tweets* não classificados que foram obtidos, após filtrar os mesmos com as seguintes palavras chave*: i)* economia; e *ii)* união europeia. Ao longo deste capítulo é identificada e representada alguma informação dos *tweets* para os passos em que este exemplo é utilizado.

4.1. Tecnologias

A escolha da linguagem de programação foi influenciada pelas *frameworks* existentes de aprendizagem automática, sendo as mais conhecidas Apache Spark e scikit-learn. A *framework* Apache Spark está disponível para diferentes linguagens de programação: Java [35], Scala [36], Python, e R [37]. A *framework* scikit-learn está disponível apenas para Python. A escolha baseou-se no facto que a *framework* Apache Spark permite criar uma aplicação escalável permitindo ser distribuída, se necessário, por diversas máquinas.

Sendo a *framework* Apache Spark multi-plataforma, a escolha da linguagem de programação ficou limitada. Sendo esta *framework* desenvolvida na linguagem Scala, fez sentido escolher esta linguagem dado ser nativa à framework. A Tabela 4.1, mostra uma breve comparação das diferenças entre as possíveis escolhas.

Dada a escolha da linguagem de programação, fica por determinar a *framework* para a aplicação web, que terá de ser numa linguagem que execute na Java Virtual Machine (JVM), como Scala ou Java. As *frameworks* web mais utilizadas [38] são Spring [39], JavaServer Faces (JSF) [40] e Google Web Toolkit (GWT) [41]. A escolhida foi Spring, dado que está bem detalhada na documentação e tem um grande apoio da comunidade de programadores web em comparação [42] com as outras opções.

Tabela 4.1: Comparação entre as linguagens de programação para a *framework* Apache Spark [43].

Comparison Points	**Java**	**Scala**	**Python**	**R**
Performance	Faster	Faster (about 10x faster than Python)	Slower	Slower
Learning Curve	Easier than Scala Tougher than Python	Steep learning curve than Java & Python	Easiest	Moderate
User Groups	Web/Hadoop programmers	Big Data Programmers	Beginners & Data Engineers	Data Scientists/ Statisticians
Usage	Web development and Hadoop Native	Spark Native	Data Engineering/ Machine Learning/ Data Visualization	Visualization/ Data Analysis/ Statistics use cases
Type of Language	Object-Oriented, General Purpose	Object-Oriented & Functional General Purpose	General Purpose	Specifically for Data Scientists. Needs conversion into Scala/Python before productizing
Concurrency	Support Concurrency	Support Concurrency	Does not Support Concurrency	NA
Ease of Use	Verbose	Lesser Verbose than Java	Least Verbose	NA
Type Safety	Statically typed	Statically typed	Dynamically Typed	Dynamically Typed
Interpreted Language (REPL)	No	No	Yes	Yes
Maturated machine learning libraries availability/ Support	Limited	Limited	Excellent	Excellent
Visualization Libraries	Limited	Limited	Excellent	Excellent

4.2. Obtenção de dados

A obtenção de dados foi realizada através da API REST do Twitter, parametrizando os pedidos realizados com a informação que o utilizador faculta à plataforma *TwtAnalysis*. Nomeadamente a língua dos *tweets* que se pretende obter e um conjunto de palavras-chave que se pretenda que os *tweets* contenham. Seguindo o exemplo referido no início deste capítulo, os parâmetros utilizados para a obtenção dos *tweets* foram os seguintes: língua com valor "pt", e palavras-chave contêm "economia" ou "união europeia". A Listagem 4.1 exemplifica o pedido realizado que satisfaz as condições acima descritas, a Listagem 4.2 apresenta a resposta ao pedido com informação dos *tweets* no formato JSON. Com esta resposta obtida são apenas aproveitados os campos enumerados anteriormente na Listagem 3.1.

Listagem 4.1: Pedido HTTP à API do Twitter para obtenção de dados.

```
GET /1.1/search/tweets.json?q
="economia" "uniao europeia"&amp;lang=pt&amp;result_type=popular HTTP/1.1
Host: api.twitter.com
Authorization: Bearer { JWT }
```

Listagem 4.2: Exemplo, simplificado, da resposta ao pedido da Listagem 4.1.


```
{
    "statuses": [
        {
            "created_at": "Sex Jun 19 12:20:03 +0000 2020",
            "id": 123456700,
            "id_str": "123456700",
            "full_text": "A chanceler alemã, Angela Merkel, falou sobre a economia do pós Bre
xit.",
            "metadata": {
                "iso_language_code": "pt",
                "result_type": "popular"
            },
            "source": "<a href=\"https://www.twitter.com\" rel=\"nofollow\">Twitter</a>",
            "user": {
                "id": 1234,
                "id_str": "1234",
                "name": "Name",
                "screen_name": "ScreenName",
                "location": "Porto, Portugal",
                "description": "Don't be evil!",
                "url": "https://t.co/",
                "protected": false,
                "created_at": "Thu Apr 29 18:06:01 +0000 2010",
                "geo_enabled": true,
                "verified": true,
                "lang": null
            },
            "lang": "pt"
        }
    ],
    "search_metadata": {
        "completed_in": 0.038,
        "max_id": 0,
        "max_id_str": "0",
        "next_results": "?max_id=00000&q=%22economia%22%20%22uniao%20europeia%22&lang=pt&incl
ude_entities=1&result_type=popular",
        "query": "%22economia%22+%22uniao+europeia%22",
        "count": 15,
        "since_id": 0,
        "since_id_str": "0"
    }
}
```

Devido ao tipo da licença de acesso à API do Twitter, versão *standard*, esta apenas disponibiliza a cada 15 minutos 450 pedidos para o *endpoint* de pesquisa de *tweets*.

Existem diversas bibliotecas disponíveis para a realização de pedidos *HyperText Transfer Protocol* (HTTP), foram, no entanto, encontradas as seguintes bibliotecas em que a sua função é facilitar o acesso à API do Twitter:

- Twitter4j [44] é uma biblioteca *open-source* e disponibiliza uma fácil integração com aplicações em Java e Scala. Disponibiliza a obtenção dos *tweets* por pedidos à API do Twitter, por outro lado possibilita também a utilização da API de *streaming* do Twitter obtendo assim os *tweets* em tempo real. Tem incorporado suporte para autorização através de OAuth [45].
- JTwitter [46] é uma biblioteca *open-source*, cujo propósito é facilitar o acesso à API do Twitter para aplicações Java e Scala. A biblioteca é disponibilizada através de um *Java Archive* (JAR).

Para realizar os pedidos à API do Twitter foi utilizada a biblioteca Twitter4j cujas funcionalidades se encontram descritas na secção 2.1. Foi escolhida esta biblioteca pois possui documentação detalhada e encontra-se atualizada com a API do Twitter.

4.3. Tratamento dos dados

Neste módulo é realizada a filtragem do conteúdo dos *tweets*. A Figura 4.2 apresenta o *workflow* das etapas necessárias para a filtragem do texto de um *tweet.*

Como detalhado na secção 3.2, é necessária a remoção de URL externos dado que não têm qualquer influência nas etapas que vão ser realizadas. No caso em que os *tweets* foram criados através do mecanismo de *retweet*, o texto do *tweet* é precedido por "RT". Dado que o texto "RT" não contribui para a classificação dos dados este deve ser retirado, acrescentando referência no modelo de dados que representa um *retweet* para que não se perca a informação que irá ser relevante para a deteção de comunidades. Ambas as filtragens são realizadas a partir de expressões *regex* [47].

É também necessária a filtragem das *stopwords*, e dado a *framework* Spark já incluir diversas bibliotecas de apoio a aprendizagem automática, tirou-se partido da biblioteca Mlib [48], que disponibiliza a lista de *stopwords* por língua e assim permite filtrar o texto dos *tweets*.

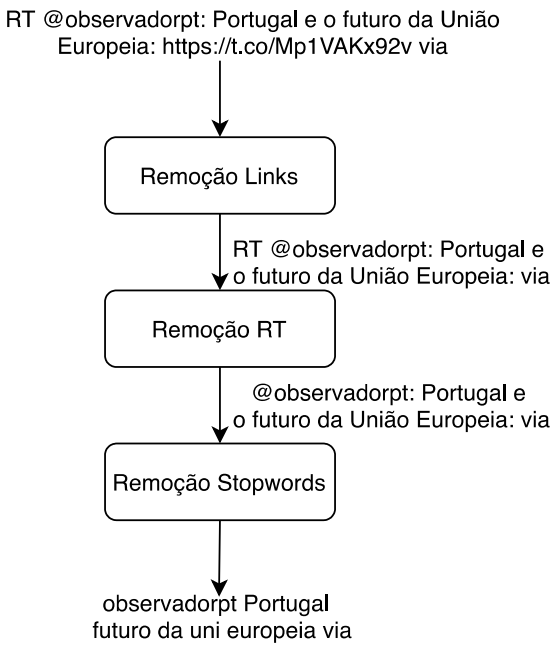


Figura 4.2: *Workflow* das etapas de tratamento de dados.

4.4. Classificação

O módulo de classificação é responsável pela classificação dos *tweets*. O diagrama apresentado na Figura 4.3 ilustra o módulo de classificação, contemplando os dados de entrada e os resultados, assim como as técnicas utilizadas na sua implementação que vão ser descritas nesta secção.

Existem dois fluxos possíveis dentro deste módulo, em que os dados de entrada e os resultados variam conforme a fase em execução: *i)* fase de treino; e *ii)* fase de generalização. Na fase de treino o módulo recebe um conjunto de *tweets* com a respetiva classificação associada e, portanto, o resultado do módulo é o modelo de classificação criado. Na fase de generalização, o módulo recebe *tweets* a serem classificados, tendo como resultado estes classificados.

Embora o módulo esteja dividido em dois fluxos, ambos têm em comum a forma como as características são tratadas. Para transformar o texto dos *tweets* em características compatíveis com o *input* dos algoritmos, foram utilizados métodos da biblioteca MLlib denominados por transformadores e estimadores. Por transformadores entenda-se um algoritmo que transforma um *DataFrame* [49], conjunto de dados que pode ser de vários tipos, noutro *DataFrame*. Enquanto que por estimadores entenda-se um algoritmo que pode ser treinado por um *DataFrame* e que produz um transformador.

Portanto, para conseguir obter a transformação das características adotou-se a abordagem referida na secção 3.3, tendo sido utilizados os transformadores *HashingTF* [50] e IDF [50]. Estes transformadores utilizados em conjunto têm como *input* um conjunto de palavras que representam um documento a ser vetorizado, e transforma-as num conjunto de vetores que representam o texto dos vários *tweets*. É o *output* destes transformadores

que fornece os dados para o processamento em ambos os fluxos, como vai ser descrito nas secções 4.4.1 e 4.4.2.

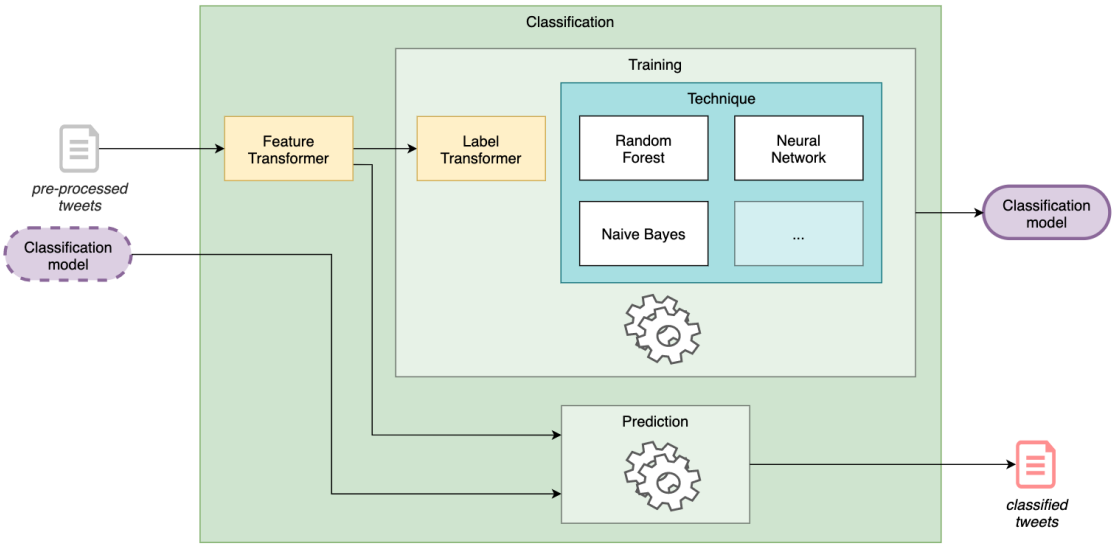


Figura 4.3: Módulo da classificação.

4.4.1. Fase de Treino

Na fase de treino o módulo recebe um conjunto de dados, nomeadamente um conjunto de *tweets*, facultado pelo utilizador. Os dados possuem um conjunto de características para caracterizar o *tweet*. Sendo estes dados de treino, possuem também as respetivas categorias.

O processamento das características encontra-se descrito no início desta secção, 4.4, devido a ser semelhante em ambas as fases. No entanto, e como mencionado na secção 3.3, é necessária a transformação das categorias do conjunto de dados que irá produzir o modelo, caso estas se encontrem em texto. Para a transformação das categorias de classificação utilizou-se um transformador, *StringIndexer* [51] que transforma as *strings*, neste caso as categorias de classificação, em índices.

Continuando com o exemplo das secções anteriores, demonstra-se como é realizada a transformação tanto das características, como das categorias. O exemplo de *workflow* de etapas para chegar a esta transformação encontra-se ilustrado na Figura 4.4. As características são o resultado da transformação já descrita e a categoria é também o resultado da aplicação do *StringIndexer*.

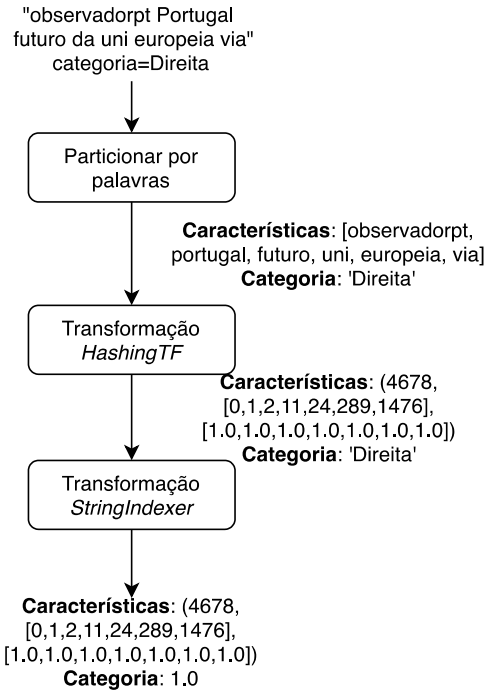


Figura 4.4: *Workflow* da transformação de características e categoria de um *tweet.*

Um dos aspetos que é necessário ter em consideração, é o balanceamento das categorias que existem no conjunto de dados que foi fornecido para criar o modelo de classificação. Pois, se o conjunto de dados possuir quantidades pouco dispersas da representação das categorias nele existentes, digamos por exemplo 80% do conjunto de dados ser categorizado como A e apenas 20% dos dados ser caracterizado como B, assim existe um enviesamento quando o modelo criado for classificar um novo conjunto de dados. Neste caso poderá classificar de forma errada como categoria A apenas porque teve mais exemplos desta categoria na sua criação.

Na plataforma *TwtAnalysis* optou-se por deixar ao critério do utilizador a técnica de classificação assim como os seus parâmetros. Como tal, recorreu-se à biblioteca MLlib do Apache Spark que possui implementações de várias técnicas de classificação, técnicas estas que são as seguintes: *i) Naive Bayes*; *ii) Random Forest*; e *iii) Neural Networks*. Estas implementações produzem um modelo capaz de classificar novos dados. Para tal é possível o utilizador parametrizar os algoritmos de forma a obter melhores resultados no seu caso de estudo. Na Tabela 4.2 encontra-se descrito cada parâmetro que deve ser ajustado nos algoritmos utilizados.

Tabela 4.2: Parâmetros dos algoritmos de classificação.

Algoritmo	Parâmetro	Tipo	Descrição
Naive Bayes	λ	*Float*	Previne que a probabilidade condicional de uma categoria que ainda não tenha sido observada seja zero. *Default*: 1.0
Random Forest	*numTrees*	*Integer*	Número de árvores na *random forest*.
	maxDepth	*Integer*	Profundidade máxima a utilizar na construção de cada árvore na *random forest*.
Neural Networks	*layers*	*Array [Integer]*	Número de valores de *input*, tamanho das *hidden layers* e tamanho da *output layer*, que deve corresponder ao número de categorias.
	stepSize	*Float*	Taxa de aprendizagem do algoritmo. *Default*: 0.03
	solver	*String*	Especifica a função de otimização. Ex: *gradient descent*. *Default*: "*l-btgs*"

Para a criação do modelo, são utilizadas as características e categorias, dos *tweets* do conjunto de dados previamente classificados, anteriormente transformadas para poderem ser processadas. Com a criação do modelo de classificação, é necessário guardá-lo para que este possa ser usado quando for necessário classificar um novo conjunto de dados. O modelo está a ser guardado de forma persistente, localmente, utilizando o método disponibilizado para este efeito pelo Apache Spark.

4.4.2. Fase de classificação

Na fase de classificação, o conjunto de dados fornecidos a este módulo são *tweets* que não se encontram classificados. Para que esta fase possa decorrer, anteriormente já deve

ter existido uma fase de treino para que exista assim um modelo de classificação a ser utilizado.

É necessário realizar o tratamento das características dos dados para que estas possam ser utilizadas com o modelo de classificação. Com as características dos dados já transformadas, é possível utilizar o modelo de classificação a aplicar aos dados em questão. Neste caso, o texto dos *tweets* são as características a serem utilizadas. Obtido o conjunto de dados categorizados, é possível utilizar essa informação para atribuir a um utilizador, a categoria mais frequente entre os *tweets* por si realizados.

4.5. Deteção das comunidades

Além da possibilidade da classificação dos dados, a plataforma *TwtAnalysis* possui também suporte para realizar a deteção de comunidades de utilizadores, que se podem extrair da interação que os mesmos realizam através dos *tweets*, nomeadamente os criados através do mecanismo de *retweet*. No Twitter existe uma relação de interação subjacente que passa pelo conceito de "seguidor" de outro utilizador, mas esta relação não é necessariamente um indicativo de influência. Quando se observa o mecanismo de *retweet* existe uma maior relação de influência entre os utilizadores [52]. Como tal, é útil inferir as comunidades existentes, de acordo com a influência verificada entre os utilizadores.

Para realizar o funcionamento descrito anteriormente, foi criado um módulo com a funcionalidade explícita de realização da deteção de comunidades. Na implementação deste módulo continuou-se a utilizar a *framework* Apache Spark pois além de técnicas de classificação de dados também possui suporte para técnicas de *clustering* dos dados.

Neste módulo é necessária a existência de um tratamento prévio dos dados antes de se proceder à deteção das comunidades. Para realizar este tratamento é utilizado o módulo de tratamento de dados descrito na secção 4.3. No entanto é necessário filtrar também os *tweets* deixando apenas os que foram criados através do mecanismo de *retweet*. Como tal, todos os outros *tweets* podem ser descartados. Tendo apenas os *retweets*, é realizada a separação dos utilizadores envolvidos nesse *retweet*, ou seja, o utilizador que produziu o *tweet* original e o utilizador que fez *retweet* ao mesmo.

Na plataforma *TwtAnalysis* optou-se por deixar ao critério do utilizador a técnica de *clustering* e seus parâmetros para a deteção de comunidades. Como tal, e tirando partido da biblioteca MLlib do Apache Spark, utilizou-se implementações de várias técnicas de *clustering*, técnicas estas que são as seguintes: *i*) *k-Means*; e *ii*) *Gaussian Mixture*. Adicionalmente, tirou-se partido de uma implementação *open-source* da técnica de *Louvain* [53] dado que a plataforma não possui nenhuma implementação. Estas implementações têm como resultado agrupamentos de utilizadores, para tal é possível o utilizador parametrizar os algoritmos para obter os melhores resultados possíveis de forma a adequarem-se ao seu caso de estudo. Na Tabela 4.3 encontra-se descrito cada parâmetro dos algoritmos utilizados.

Tabela 4.3: Parâmetros dos algoritmos de deteção de comunidades.

Algoritmo	Parâmetro	Tipo	Descrição
k-Means	*k*	*Integer*	Número de *clusters* em que os dados vão ser divididos.
	maxIterations	*Integer*	Máximo número de iterações permitidas.
Gaussian Mixture	*k*	*Integer*	Número de *clusters* em que os dados vão ser divididos.
	maxIterations	*Integer*	Máximo número de iterações permitidas.
Louvain	*minimum compression progress*	*Integer*	Número mínimo de nós que causaram alterações na comunidade entre passagens.

Para efeitos de ilustração, considere que no módulo de classificação foi utilizado um conjunto de dados do projeto HATE, como treino, que originou um modelo de classificação. Este modelo de classificação foi utilizado para classificar um outro conjunto de *tweets* sem categorização. Assuma o conjunto de *tweets* apresentados na Tabela 4.4. De forma a facilitar a interpretação dos dados, utilizou-se como simbologia do texto do *tweet* `Tx`, e para os *retweets* `RT Tx`.

Na Tabela 4.5 encontram-se os *tweets* da Tabela 4.4, que foram realizados através do mecanismo de *retweet*. Após a utilização do módulo descrito nesta secção sobre o conjunto de *retweets* obteve-se as comunidades indicadas na coluna "Comunidade".

Tabela 4.4: Conjunto de dados exemplo classificados.

Id Twitter	Id User	Text	Classificação do tweet
1	1	`T1`	Direita
2	2	`T2`	Esquerda
3	3	`T3`	N/D
4	4	`T4`	Direita
5	5	`T5`	Esquerda
6	5	`T6`	Esquerda
7	4	`RT T1`	Direita
8	4	`RT T2`	Esquerda
9	5	`RT T2`	Esquerda
10	6	`RT T3`	N/D

Tabela 4.5: Conjunto de *retweets* dos dados exemplo classificados e agrupados.

Id Twitter	Id User	Text	Classificação do tweet	Comunidade
7	4	`RT T1`	Direita	1
8	4	`RT T2`	Esquerda	1
9	5	`RT T2`	Esquerda	1
10	6	`RT T3`	N/D	2

Com estes dados é possível verificar que existiram três *tweets* que estão na origem destes *retweets*. O diagrama da Figura 4.5, representa a informação que se consegue obter da classificação dos *tweets* e das comunidades detectadas. Neste diagrama, os utilizadores que publicaram os *tweets* originais aos quais foi dado *retweet* foram o User 1, User 2 e User 3. E como referido anteriormente, existem três *tweets* que estiveram na origem destes *retweets*. Os *tweets* encontram-se classificados com a categoria predita no módulo de classificação e cada utilizador encontra-se também categorizado com a categoria mais frequente no conjunto de *tweets* que ele publicou.

Assim, consegue-se categorizar as comunidades encontradas, pela categoria mais frequente dos *tweets* dos utilizadores que a constituem. Deste modo, a Comunidade 1 encontra-se classificada com a categoria "Esquerda", pois o conjunto de utilizadores pelos quais é constituída tem a categoria "Esquerda" como mais frequente. E a Comunidade 2 encontra-se classificada com a categoria "N/D" pois apenas possui utilizadores que foram categorizados com "N/D", sendo por consequência esta a categoria mais frequente.

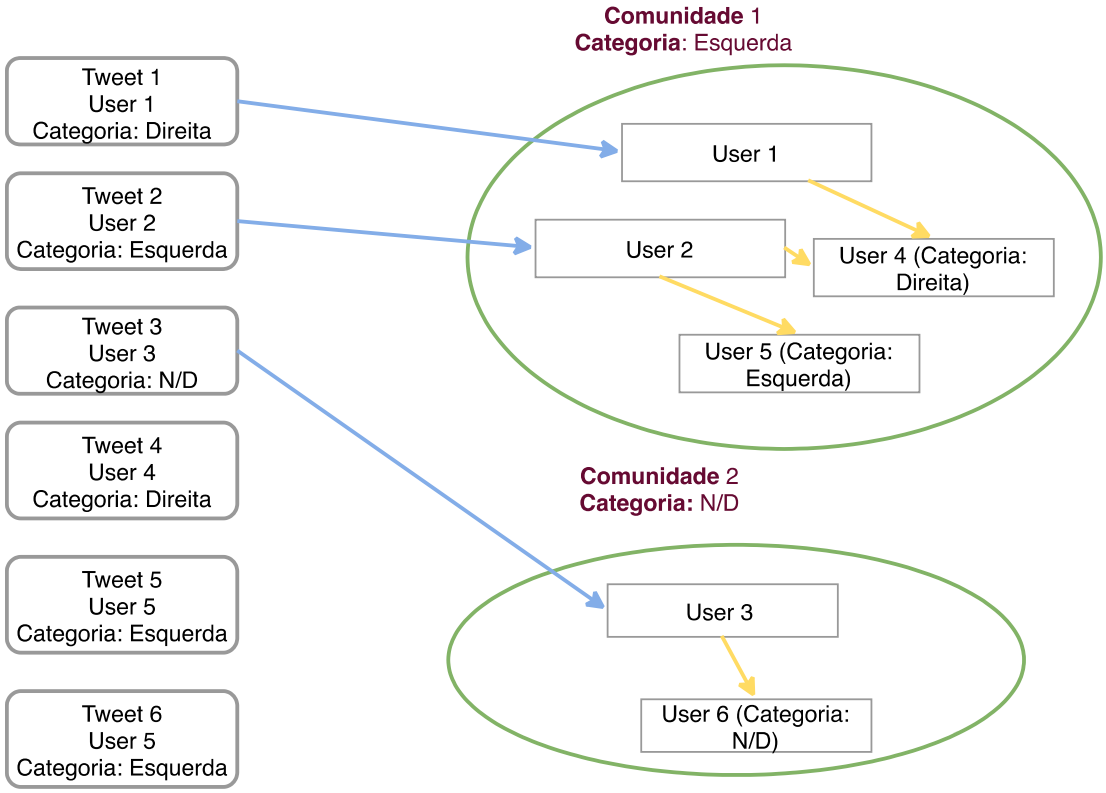


Figura 4.5: Diagrama exemplo de agregação dos resultados de classificação e deteção de comunidades.

4.6. *TwtAnalysis*

Criados os módulos descritos neste capítulo, desenvolveu-se uma plataforma com o intuito de agregar as funcionalidades disponibilizadas por cada um dos módulos. A plataforma *TwtAnalysis* é apresentada ao utilizador através de uma aplicação *web*.

A plataforma foi implementada utilizando a linguagem de programação Java e utilizou-se a *framework* Spring MVC [54]. A plataforma disponibiliza funcionalidades distintas, nomeadamente obtenção de um modelo de classificação, classificar novos conjuntos de *tweets*, agrupá-los em comunidades de utilizadores e disponibiliza uma análise estatística dos resultados obtidos da classificação e da deteção de comunidades. As secções seguintes apresentam o uso pretendido desta plataforma.

4.6.1. Obter modelo de classificação

Como ilustrado na Figura 4.6, esta funcionalidade está exposta através da receção dos seguintes parâmetros por parte do utilizador: *i)* ficheiro CSV com os *tweets* previamente classificados; *ii)* nome da coluna do ficheiro onde se encontram as classificações dos *tweets*; *iii)* nome da coluna do ficheiro onde se encontra o texto do *tweet*; *iv)* algoritmo a utilizar para construir o modelo de classificação; e *v)* parâmetros para o algoritmo escolhido a utilizar na classificação.

Os parâmetros *ii)* e *iii)* facultados pelo utilizador são aplicados, para obter as colunas que correspondem ao texto do *tweet* e às classificações. Sendo os seus valores utilizados para obter o modelo de classificação, como características e classificações, respetivamente. O conteúdo do parâmetro *i)* facultado pelo utilizador é usado como *input* do módulo de classificação para realizar a criação de um modelo de classificação como referido na secção 4.4.1. Os parâmetros *iv)* e *v)* são utilizados para definir o algoritmo de classificação que a plataforma irá utilizar assim como os seus parâmetros.

Assumindo que o parâmetro *ii*) tem o valor "`orient`" e o parâmetro *iii*) o valor "`text`". O ficheiro que a plataforma espera deve ter um cabeçalho com o nome das colunas, podendo ter *n* linhas e *m* colunas. As colunas "`orient`" e "`text`" têm de fazer parte do conjunto das colunas, como apresentado na Listagem 4.3.

Listagem 4.3: Ficheiro de *input* exemplo.

```
1. tweet_id    text    orient  populismo
2. 991631251  Preciso de um emprego   0   9
3. 991433104  Deixem.se de coisas.. que nenhuma empresa pode produzir se não vender a produção. O que
    gera emprego, é o consumo . 0   9
4. 991511680  Acordar amanhã com a notícia de que fui chamado p um emprego seria um sonho 0   9
5. 991654033  Enfim, um emprego para poder pagar o calote.    1   9
6. 103118051 Macron: um presidente pós-União Europeia https://t.co/  1   9
```

Uma vez criado o modelo, este é guardado pela plataforma e é disponibilizado um identificador único, como ilustrado na Figura 4.6. Desta forma, possibilita-se a sua utilização em futuras predições feitas na plataforma *TwtAnalysis*. Além do identificador do modelo é também disponibilizada a *accuracy* e a matriz de confusão que o modelo produziu. Tendo para isso o conjunto de dados de treino ter sido particionado com 80%-20% para utilizar 80% dos dados como treino e os restantes 20% para testar o modelo e daí obter as métricas disponibilizadas ao utilizador.

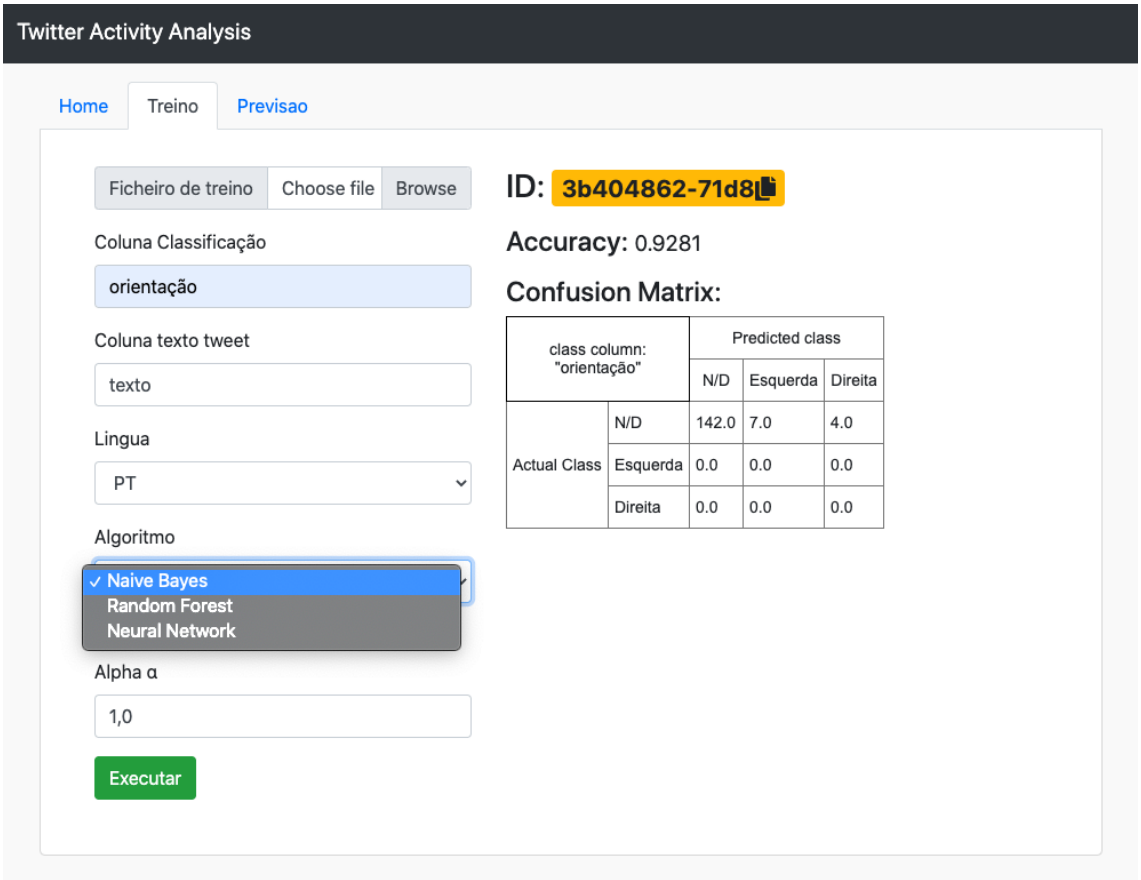


Figura 4.6: Exemplo de utilização da plataforma *TwtAnalysis* para obtenção do modelo de classificação.

4.6.2. Efetuar predições

A plataforma pode também ser utilizada para realizar predições, cujos passos são: a classificação, o detetar das comunidades e realizar uma análise estatística sobre as classificações obtidas assim como sobre o resultado das comunidades encontradas.

Para tal o utilizador tem de fornecer os seguintes parâmetros: *i)* identificador do modelo a ser utilizado para a classificação; *ii)* palavras-chave, separadas por vírgula, que os *tweets* devem conter; *iii)* língua na qual os *tweets* devem ter sido realizados; *iv*) algoritmo a ser utilizado para a deteção de comunidades; e *v)* parâmetros do algoritmo escolhido no parâmetro anterior.

O parâmetro *i)* é utilizado para obter o modelo previamente criado. Os parâmetros *iv)* e *v)* são utilizados para definir qual o algoritmo a utilizar, e quais os seus parâmetros, para realizar a deteção de comunidades. Os restantes parâmetros, *ii)* e *iii)*, facultados pelo utilizador, são utilizados na obtenção dos *tweets* através da API do Twitter. Para recolher os dados recorre-se ao módulo de obtenção de dados que foi descrito na secção 4.2.

Uma vez tendo os dados a utilizar, é seguido o *workflow* de etapas necessárias para obter uma análise sobre os dados, etapas estas que já não presumem interação com o utilizador e são internas ao sistema. Sendo estas: *i)* tratamento de dados; *ii)* classificação; e *iii)*

deteção de comunidades, que estão disponibilizadas através de módulos que se encontram descritos nas secções anteriores. Por fim, após o *workflow* de etapas terminar a execução, são disponibilizados ao utilizador os resultados obtidos, ou seja, as categorias e as comunidades encontradas assim como uma análise estatística sobre os *tweets* categorizados e sobre as comunidades encontradas que já se encontram categorizadas. Além dos resultados obtidos da classificação e deteção de comunidades é também disponibilizada a métrica *silhoutte* sobre as comunidades encontradas com o algoritmo e parâmetros utilizados que o utilizador escolheu. É também possível o utilizador fazer *download* de um ficheiro que contém os *tweets* classificados e no caso de serem *retweets*, a comunidade a que o utilizador que realizou o *retweet* pertence. Na Figura 4.7 encontram-se os *tweets* classificados do exemplo concreto que tem vindo a ser descrito.

Combinando as categorias com as comunidades encontradas, para o exemplo concreto de aferir a orientação política de um conjunto de *tweets*, é possível disponibilizar ao utilizador a informação contida na Figura 4.8.

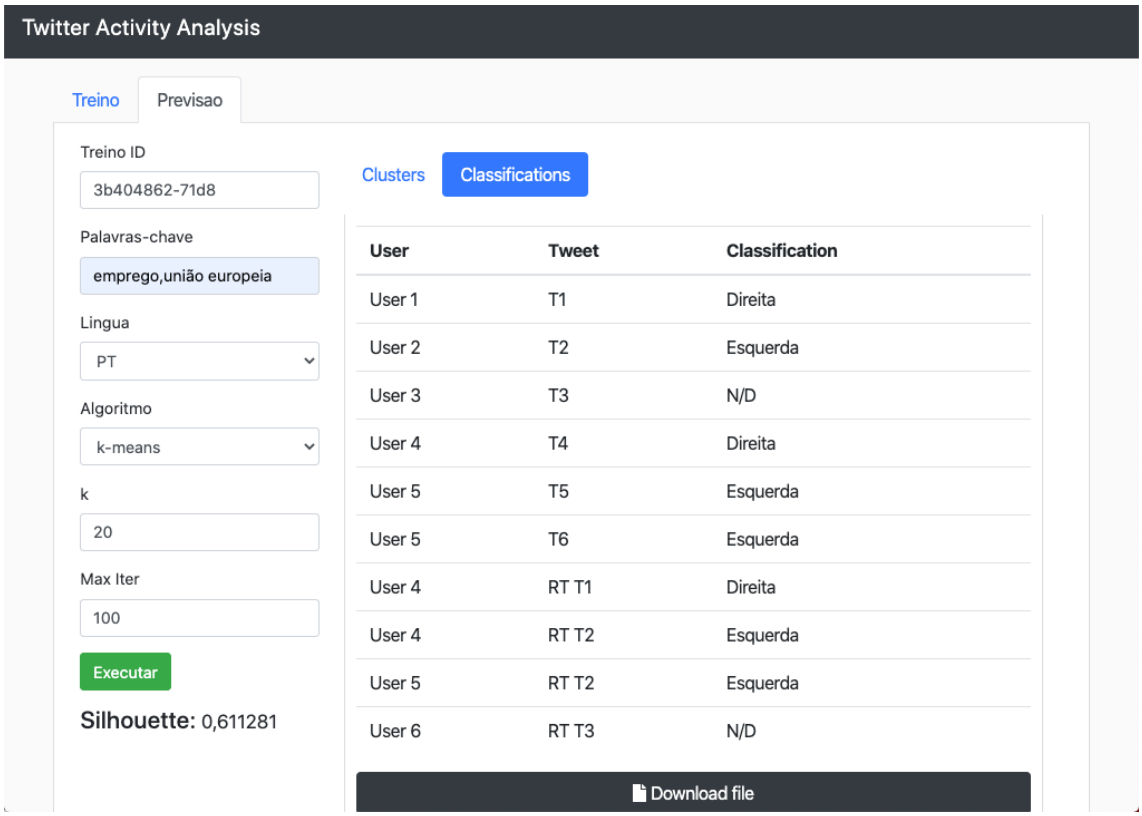


Figura 4.7: Exemplo da utilização da plataforma *TwtAnalysis* para a obtenção de predições sobre a categorização dos *tweets*.

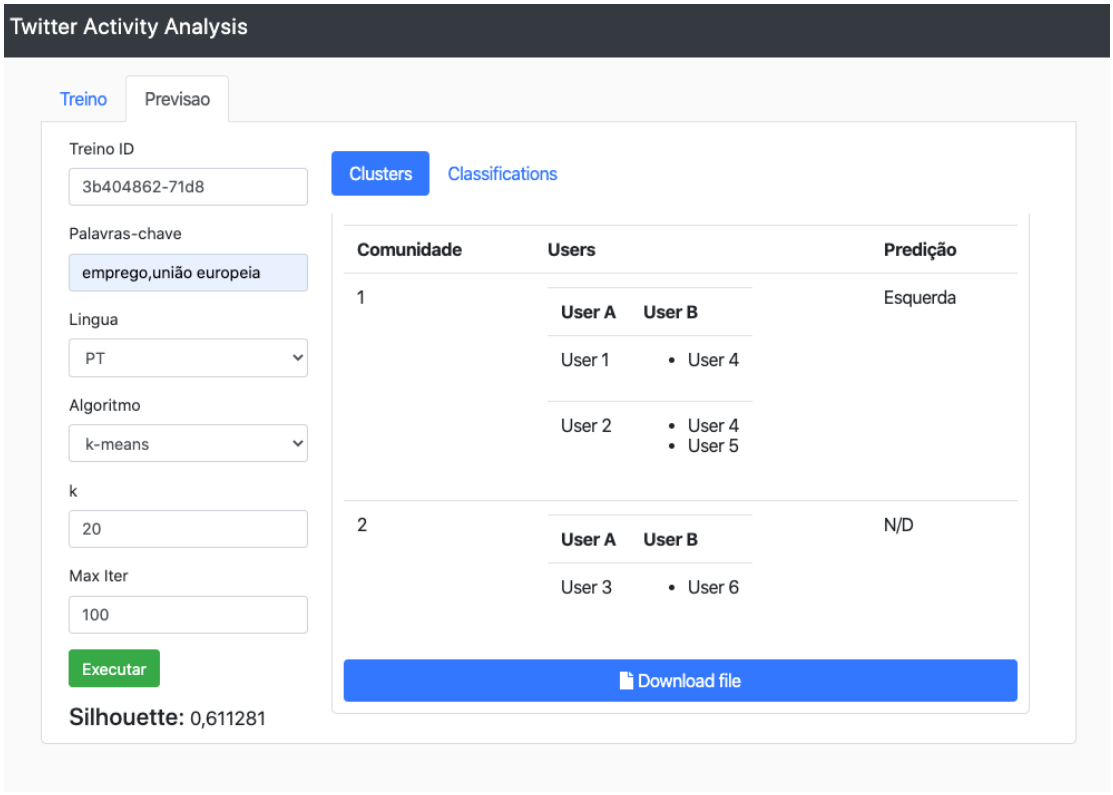


Figura 4.8: Exemplo da utilização da plataforma *TwtAnalysis* para a obtenção de predições sobre as comunidades detetadas nos *retweets*.

5. Avaliação experimental

A plataforma *TwtAnalysis* desenvolvida no contexto deste projeto utiliza diversas técnicas de forma a obter a categorização e agrupamento dos dados. Neste capítulo é feita uma avaliação, utilizando procedimentos e métricas, para se tentar comparar os resultados obtidos pelas diferentes técnicas disponíveis na plataforma.

O capítulo encontra-se dividido em cinco secções. A secção 5.1 caracteriza os processos para obter os resultados. Na secção 5.2 encontra-se a caracterização dos conjuntos de dados a utilizar na avaliação experimental. A secção 5.3 descreve os procedimentos e métricas utilizadas e compara as técnicas utilizadas para a categorização dos dados. Na secção 5.4 encontram-se descritos os procedimentos e métricas utilizadas para o agrupamento de dados. A secção 5.5 apresenta conclusões sobre os resultados obtidos e como estes se refletem na plataforma desenvolvida.

5.1. Metodologia

Para a execução dos testes apresentados neste capítulo, foi utilizada uma máquina que dispõe de um processador *Intel Core* i5 de 2.7GHz, com 8 GB de memória RAM DDR3. Utilizou-se uma única instância de *Spark*, a executar localmente, para a realização dos testes.

Realizaram-se os testes pelos módulos desenvolvidos: classificação e deteção de comunidades. Para a classificação utilizaram-se diferentes conjuntos de dados e técnicas de classificação, de forma a obter diferentes métricas para os comparar. Para a deteção de comunidades utilizou-se um conjunto de dados e diferentes técnicas de *clustering* mas desta vez apenas se considerou os *retweets*. As secções seguintes descrevem o conjunto de dados utilizado e as comparações entre as técnicas utilizadas.

Na plataforma desenvolvida, sempre que o utilizador realiza o treino com um algoritmo de classificação e respetivos parâmetros à sua escolha, é disponibilizada a *accuracy* e a matriz de confusão que o modelo produz. Foram realizados no contexto desta avaliação experimental outras métricas de forma a ter mais sensibilidade se existe alguma técnica de classificação que é sempre melhor que as restantes disponibilizadas.

5.2. Conjunto de dados

Ao longo desta secção vão ser descritos os conjuntos de dados a utilizar tanto na avaliação das técnicas de classificação como nas técnicas de *clustering* disponibilizadas pela plataforma desenvolvida.

5.2.1. Conjunto de dados do projeto HATE

Os *tweets* deste conjunto de dados foram obtidos entre Maio e Julho de 2018, e as palavras-chave para a sua obtenção foram: *i)* economia; e *ii)* união europeia. Estes foram depois categorizados manualmente dado o conteúdo do texto. Na Tabela 5.1 encontra-se a caracterização deste conjunto de dados, relativamente aos campos que o constituem, os possíveis valores que podem tomar e uma breve descrição do que significam.

Tabela 5.1: Caracterização do conjunto de dados do projeto HATE.

Campo	**Tipo**	**Valor**	**Descrição**
tweet_id	*Float*	-	Identificador do *tweet*
text	*String*	-	Texto do *tweet* a analisar
orient	*Integer*	{0,1,9}	Orientação política do *tweet*. 0, de direita; 1, de esquerda; ou 9, não foi possível determinar (N/D)
populism	Integer	{0, 1}	Populismo do *tweet.* 0, tenta apelar ao "povo"; 1, não existe evidências de apelo ao "povo"

No contexto desta avaliação experimental a coluna referente à orientação política, é tomada em consideração para avaliar os *tweets*, ou seja, corresponde à sua categorização.

Como se pode verificar na Tabela 5.2, o conjunto de dados é constituído por 856 *tweets* classificados, dos quais 3,27% foram categorizados previamente como sendo de "Direita", 5,38% como sendo de "Esquerda" e os restantes 91,35% não foram possíveis determinar a orientação política, "N/D".

O conjunto de dados possui outros campos que não foram considerados relevantes para esta avaliação experimental, como tal apenas foram utilizadas as colunas cujos campos se encontram a negrito na Tabela 5.1.

Tabela 5.2: Distribuição das categorias no conjunto de dados do projeto HATE.

Número de *tweets* classificados	**0 (Direita)**	**1 (Esquerda)**	**9 (N/D)**
855	3,27%	5,38%	91,35%

5.2.2. Conjunto de dados *Brexit* [55]

Este conjunto de dados foi escolhido por ter uma proporção entre categorias com maior uniformidade. O conjunto de dados foi categorizado manualmente, e é constituído por *tweets* de 449 membros do parlamento do Reino Unido em 2016. *Tweets* que continham um conjunto de palavras-chave associadas à união europeia e ao *brexit*.

Na Tabela 5.3 encontra-se sumarizada a informação dos campos que constituem o conjunto de dados, encontrando-se a negrito os campos utilizados na classificação.

Como se pode observar na Tabela 5.4 este conjunto de dados é constituído por 53576 *tweets* classificados, sendo que 66,03% destes foram categorizados previamente em como o seu utilizador queria ficar na união europeia (*stay*) e 33,97% foram categorizados previamente como que queria sair (*leave*).

Tabela 5.3: Caracterização do conjunto de dados *Brexit*.

Campo	**Tipo**	**Valores**	**Descrição**
name	*String*	-	Nome do utilizador
screen_name	*String*	-	Nome do utilizador no Twitter
id	*Integer*	-	Identificador do *tweet*
created_at	*DateTime*	-	Data de criação do *tweet*
text	*String*	-	Texto do *tweet*
label	*String*	{Stay, Leave}	Categoria do *tweet*, ponto de vista político em relação ao Brexit. Sair ou Ficar.

Tabela 5.4: Distribuição de categorias do conjunto de dados *Brexit*.

Número de *tweets* classificados	**Stay**	**Leave**
53576	66,03%	33,97%

5.3. Classificação

Para avaliar os modelos criados pelas técnicas que permitem a categorização dos dados utilizou-se um procedimento de reamostragem denominado *Cross Validation* [56]. Este procedimento consiste em aleatoriamente dividir o conjunto de dados em k grupos (k é facultado à priori) e iterativamente para cada grupo utiliza-o como conjunto de dados de teste e os restantes k-1 grupos como conjunto de dados de treino do modelo. Utilizou-se *Cross Validation* com k=5 e utilizou-se uma percentagem de 80% do conjunto de dados para encontrar o melhor modelo utilizando o procedimento de reamostragem descrito deixando 20% para validar os resultados produzidos recorrendo a um conjunto de métricas.

As métricas utilizadas foram *accuracy* que traduz o rácio do número de predições corretas pelo número total de predições que foram realizadas. Outra métrica utilizada foi *F-measure*, tomando valores entre [0,1] traduz quão preciso o modelo é, assim como, quão robusto o modelo é. Esta métrica é calculada utilizando a *precision* e *recall*. Traduzindo-se a *accuracy*, a *precision*, o *recall* e a *f-measure* pelas seguintes expressões:

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives}. \quad (22)$$

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}. \quad (23)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}. \quad (24)$$

$$F\ Measure = 2 * \frac{1}{(\frac{1}{Precision} + \frac{1}{Recall})}. \quad (25)$$

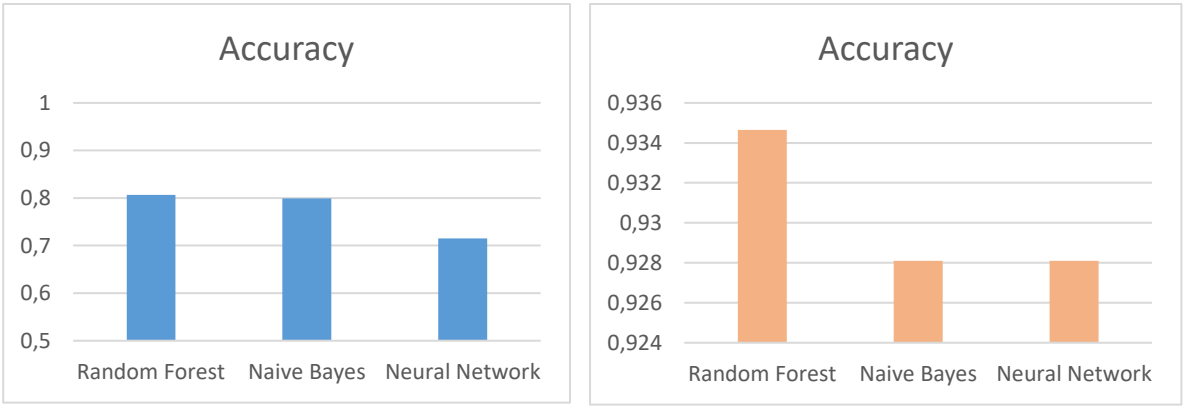
Aplicando os procedimentos e utilizando as métricas descritas obteve-se os resultados da Tabela 5.5 para o conjunto de dados *Brexit*. E os resultados da Tabela 5.6 para o conjunto de dados do projeto HATE. A Figura 5.1 apresenta dois gráficos de uma das métricas obtidas, *accuracy*. É possível verificar-se que o gráfico de dados do projeto HATE tem valores similares em todas as técnicas e valores mais altos que o conjunto de dados do *Brexit*. Esta diferença advém do facto de o conjunto de dados não ter uma distribuição uniforme das categorias, daí também a sua *accuracy* ser alta no conjunto de dados HATE. Pois existindo 91,35% de casos com a categoria "N/D" os modelos criados ficam *overfitted* acabando por prever todo ou quase todo o conjunto de dados com esta categoria.

Tabela 5.5: Resultados obtidos da avaliação das técnicas de classificação, no conjunto de dados *Brexit*.

Brexit	**Naïve Bayes**	**Random Forest**	**Neural Network**
Accuracy	0,799528	0,806307	0,715074
F-Measure	0,802473	0,802245	0,680353
Precision	0,809301	0,802132	0,701701
Recall	0,799528	0,806307	0,715074
Tempo de treino (ms)	32635	499257	685778
Tempo de predição (ms)	135	316	267

Tabela 5.6: Resultados obtidos da avaliação das técnicas de classificação, no conjunto de dados HATE.

HATE	**Naïve Bayes**	**Random Forest**	**Neural Network**
Accuracy	0,934641	0,928105	0,928105
F-Measure	0,916945	0,893497	0,893497
Precision	0,921438	0,861378	0,861378
Recall	0,934641	0,928105	0,928105
Tempo de treino (ms)	38772	12298	73872
Tempo de predição (ms)	85	76	133



(a) (b)

Figura 5.1: Graficos da métrica *Accuracy*.
(a) gráfico do conjunto de dados Brexit. (b) gráfico do conjunto de dados HATE.

Os algoritmos utilizados nesta secção fazem todos parte da biblioteca MLlib do Apache Spark. Os parâmetros utilizados para as técnicas foram fixados de acordo com a capacidade da máquina em que os testes foram realizados. Das técnicas utilizadas, conclui-se que *Random Forest* e *Naive Bayes* possuem resultados idênticos a nível de *accuracy*. Existindo diferença maioritariamente no tempo a ser realizado o treino do modelo, sendo o *Naive Bayes* mais rápido que o *Random Forest*. A técnica *Neural Networks*, para o conjunto de dados *Brexit* que era consideravelmente maior, não obteve resultados tão bons quanto as outras técnicas tendo tempo de treino consideravelmente superior do que ambas as duas técnicas avaliadas. Como tal, as técnicas mais indicadas a utilizar neste contexto, mas dependendo do conjunto de dados, poderiam ser a *Random Forest* ou *Naive Bayes*.

5.4. Deteção de comunidades

Para realizar a deteção de comunidades são apenas utilizados os *retweets* do conjunto de dados utilizados nesta avaliação experimental. Os conjuntos de dados a usar nesta secção iriam ser os mesmos que a secção anterior, no entanto o conjunto de dados do projeto HATE não disponibiliza nenhuma informação do utilizador que realizou o *tweet*, não existindo forma de associar utilizadores através do mecanismo de *retweet*. Como tal, foi utilizado apenas o conjunto de dados Brexit, em que existem 31895 *retweets*.

As métricas utilizadas foram *silhouette* e modularidade, permitindo aferir se o número de grupos foi adequado para o conjunto de dados. A métrica *silhouette* tem como intervalo de valores [-1, 1] e é utilizada de forma a medir o quão semelhante um elemento do grupo é com os restantes pertencentes ao mesmo grupo, comparando com quão semelhante o elemento é a outros grupos que não o seu. Esta medida foi calculada utilizando a distância euclidiana e permite obter o melhor número de grupos produzidos, maximizando a sua semelhança interna (no mesmo grupo) face à semelhança a elementos de outros grupos. A métrica modularidade tem como valores [-1/2, 1] permite medir a força da divisão da rede em grupos. Tal que, redes com maior modularidade possuem conexões mais fortes com os nós do mesmo grupo, mas menores conexões com os nós de outros grupos que não o seu.

Obteve-se os resultados ilustrados na Tabela 5.7 para as diferentes técnicas de agrupamento para os *retweets* existentes do conjunto de dados *Brexit*. Foi utilizada a métrica *silhouette* para todas as técnicas pois esta é a única métrica que a *framework* Apache Spark disponibiliza. Como a técnica *Louvain* foi obtida de uma implementação de uma biblioteca externa, dado o Apache Spark não ter implementação, esta biblioteca disponibiliza a métrica modularidade. Não foram encontradas bibliotecas disponíveis de forma a avaliar segundo a modularidade para as técnicas *k-means* e *Gaussian Mixture*.

Tabela 5.7: Resultados obtidos da avaliação das técnicas de agrupamento.

	k-means	**Gaussian Mixture**	**Louvain**
Silhouette	0,611282112	-0,034689928	-0,631193475
Modularity	-	-	0,237459629
Tempo (ms)	51948	68402	1292804

A técnica *Louvain* é a que demora mais tempo, por não estar otimizada para a *framework* Apache Spark como as outras técnicas disponibilizadas. A única métrica disponível para comparar as técnicas utilizadas é a *silhouette*, no entanto esta pode não ser a mais adequada para avaliar as técnicas de deteção de comunidades disponíveis na plataforma desenvolvida.

5.5. Conclusões experimentais

Nas secções anteriores, pretendia-se comparar as diferentes técnicas disponibilizadas pela plataforma para realizar a classificação e a deteção de comunidades. Nos casos de estudo utilizados observou-se que as técnicas *Naive Bayes* e *Random Forest* tiveram um melhor comportamento para a classificação. Na deteção de comunidades e utilizando a única métrica disponível, observou-se que para o caso de estudo aplicado a técnica *k-Means* teve melhor comportamento.

No entanto o desempenho das diversas técnicas varia conforme o caso de estudo sobre o qual são aplicadas, como foi possível verificar das diferenças entre o conjunto de dados HATE e *Brexit* na classificação, assim como da capacidade computacional existente. Dado que a plataforma desenvolvida, define um fluxo de processamento a realizar não estando vinculada a um conjunto de dados específico, cabe ao utilizador escolher a técnica e os parâmetros que pretender de forma a se adequarem ao seu caso de estudo. Na plataforma desenvolvida estão disponíveis todas as técnicas mencionadas neste capítulo.

6. Conclusões

Tem vindo a existir um aumento na utilização de redes sociais, nos últimos anos. Este aumento deve-se em grande parte ao avanço tecnológico na área dos dispositivos móveis e à disponibilização das redes sociais por praticamente todos os utilizadores à distância de um *click* no seu *smartphone*. Assim existiu também um aumento da quantidade de dados que nestas redes sociais são produzidos. Com este aumento, surgiu a necessidade de aplicar técnicas de aprendizagem automática para tirar partido dos mesmos. No entanto, as soluções que geralmente são apresentadas têm como objetivo aferir relativamente à polaridade dos dados, não tendo a possibilidade de categorizá-los de outras formas.

No trabalho desenvolvido, apresentado neste documento, o foco foi o desenvolvimento de uma plataforma que pelas suas funcionalidades tenta resolver a falta que foi notada sobre a categorização e agrupamento dos dados. No contexto da plataforma desenvolvida foi, no entanto, apenas considerada a rede social Twitter, pois decidiu-se que pelos utilizadores que também a utilizam, nomeadamente os media e figuras com grande influência na sociedade atualmente, esta seria uma boa fonte de dados para obter informações relativamente às opiniões de utilizadores. Além da categorização de dados também foi realizada a deteção das comunidades existentes utilizando os *retweets*, tendo assim a forma como os utilizadores se relacionam.

A plataforma desenvolvida é constituída por um conjunto de módulos que foram implementados de forma a que todos juntos cumprissem os requisitos que foram verificados para este projeto. Para que a plataforma suporte a categorização dos *tweets*, foi implementado todo o processo anterior até que possa utilizar os dados para a classificação, nomeadamente a obtenção de *tweets* e o tratamento dos mesmos, sendo depois aplicada uma das técnicas disponibilizadas, nesta plataforma, para proceder a esta categorização. Foi também implementada a inferência da influência dos utilizadores através dos *retweets* que realizam. Para perceber esta influência, foi feito o *clustering* aplicando uma de várias técnicas disponibilizadas nesta plataforma.

6.1. Trabalho Futuro

Existem várias melhorias e funcionalidades que poderiam ser adicionadas de forma a melhorar o projeto desenvolvido.

Uma funcionalidade a acrescentar seria a possibilidade de realizar predições com mais do que um conjunto de categorias. Por exemplo classificar um conjunto de *tweets* quanto à orientação política e quanto ao género do utilizador, de forma a aferir numa comunidade qual a orientação política predominante e se existe relação com o género dos utilizadores

que a constituem. Na deteção de comunidades poder-se-ia, na construção da rede de *retweets,* levar em consideração a quantidade de *retweets* que o utilizador A faz a *tweets* do utilizador B pois desta forma poderíamos adicionar pesos à rede, expressando assim maior grau de influência entre os dois. Ou como referido anteriormente, utilizar outro tipo de associações entre utilizadores na criação da rede, como por exemplo os seus seguidores.

Como a plataforma foi executada localmente utilizando uma instância de Apache Spark, e por efeitos de limitação da máquina, só se conseguiu testar utilizando quatro *worker nodes*. No futuro, seria interessante caso houvesse acesso, executar a plataforma num *cluster* com mais instâncias.

Outra funcionalidade da qual este projeto beneficiaria, seria da obtenção dos *tweets* para realizar predições através de *streaming*, executando todo o processo de análise existente nos diversos módulos do projeto e assim seria possível ter uma categorização dos *tweets* e deteção de comunidades em tempo real. Poderiam ser também adicionadas bibliotecas que permitissem a visualização dos dados na plataforma desenvolvida através de gráficos ou diagramas.

Na obtenção de dados também seria interessante, embora não fosse esse o foco da plataforma desenvolvida, introduzir outra fonte de dados. Como por exemplo, outra rede social.

Referências

[1] "Youtube," [Online]. Available: www.youtube.com.

[2] "Facebook," [Online]. Available: https://www.facebook.com/.

[3] Twitter, "Twitter," [Online]. Available: https://twitter.com/.

[4] S. Salgado, 1 Outubro 2018. [Online]. Available: www.hate.ics.ulisboa.pt. [Acedido em 1 Outubro 2019].

[5] "Reddit," [Online]. Available: https://www.reddit.com/.

[6] Twitter, Inc., "Twitter developer platform," [Online]. Available: https://developer.twitter.com/.

[7] H. Ishwaran, "Variable importance in binary regression trees and forests.," *Electronic Journal of Statistics,* pp. 1: 519-537, 2007.

[8] C. Piech, "K Means," [Online]. Available: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html. [Acedido em 17 Janeiro 2019].

[9] A. Géron, Hands-On Machine Learning with Scikit-Learn and TenserFlow, O'Reilly Media, Inc., 2017.

[10] S. Fortunado, "Community detection in graphs.," *Physics reports,* vol. 486, nº 3-5, pp. 79-82, 2010.

[11] K. Sirotkin e J. W. Wilbur, "The automatic identification of stop words.," *Journal of information science,* vol. 18, nº 1, pp. 45-55, 1992.

[12] P. Raghavan, H. Schütze e C. Manning, Introduction to information retrieval., Cambridge university press, 2008.

[13] A. C. Müller e S. Guido, Introduction to machine learning with Python: a guide for data scientists, O'Reilly Media, Inc., 2016.

[14] "Pandas documentation," [Online]. Available: https://pandas.pydata.org/docs/.

[15] "Python," [Online]. Available: https://www.python.org/.

[16] R. S. Sutton e A. S. Barto, Introduction to reinforcement learning., Cambridge: MIT press, 1998.

[17] R. Caruana e A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms," *Proceedings of the 23rd International conference of Machine Learning.,* pp. 161-168, 2006.

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

[19] H. Zhou, J. Chen e W. Peng, "An implementation of ID3-decision tree learning algorithm.," 2019.

[20] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, nº 2, pp. 123-140, 1996.

[21] M. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

[22] A. Arnx, “First neural network for begginers explained (with code),” 13 Janeira 2019. [Online]. Available: https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf. [Acedido em Julho 2020].

[23] A. Sharma, “Understanding Activation Functions in Neural Networks,” 30 Março 2017. [Online]. Available: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. [Acedido em Julho 2020].

[24] P. Barrett, “Euclidean distance: Raw, normalised, and double-scaled coefficients. Unpublished paper retrieved from http://www. pbmetrix. com/techpapers/Euclidean_Distance. pdf.,” 2006.

[25] “Normal Distribution - Wolfram MathWorld,” [Online]. Available: https://mathworld.wolfram.com/NormalDistribution.html. [Acedido em Julho 2020].

[26] C. M. Bishop, Pattern recognition and machine learning., Springer, 2006.

[27] V. D. Blondel, J.-L. Guillaume, R. Lambiotte e E. Lefebvre, “Fast unfolding of communities in large networks,” 25 Julho 2008.

[28] “scikit-learn Machine learning in Python,” [Online]. Available: https://scikit-learn.org/stable/.

[29] “Apache Spark,” [Online]. Available: https://databricks.com/spark/about.

[30] “Apache Spark MLlib,” [Online]. Available: https://spark.apache.org/mllib/. [Acedido em Julho 2020].

[31] “Cluster Mode Overview,” [Online]. Available: https://spark.apache.org/docs/latest/cluster-overview.html.

[32] Apache Spark, “Apache Spark - Cluster Mode Overview,” [Online]. Available: https://spark.apache.org/docs/latest/cluster-overview.html. [Acedido em Maio 2020].

[33] “https://monkeylearn.com/,” [Online]. [Acedido em Julho 2020].

[34] G. Salton, A. Wong e C. Yang, “A vector space model for automatic indexing,” *Communications of the ACM,* pp. 613-620, Novembro 1975.

[35] “Java Language and Virtual Machine Specifications,” [Online]. Available: https://docs.oracle.com/javase/specs/index.html.

[36] “The Scala Programming Language,” [Online]. Available: https://www.scala-lang.org/.

[37] “R: The R Project for Statistical Computing,” [Online]. Available: https://www.r-project.org/. [Acedido em Julho 2020].

[38] V. Power, “Most Popular Java Web Frameworks in 2019,” 23 Julho 2019. [Online]. Available: https://rollbar.com/blog/most-popular-java-web-frameworks/. [Acedido em Julho 2020].

[39] “Spring | Web Applications,” [Online]. Available: https://spring.io/web-applications. [Acedido em Julho 2020].

[40] “JavaServer Faces.org,” [Online]. Available: http://www.javaserverfaces.org/. [Acedido em Julho 2020].

[41] “[GWT],” [Online]. Available: http://www.gwtproject.org/. [Acedido em Julho 2020].

[42] “StackOverflow - Newest 'spring' Questions,” [Online]. Available: https://stackoverflow.com/questions/tagged/spring. [Acedido em Julho 2020].

[43] S. Deshpande, “Scala Vs Python Vs R Vs Java - Which language is better for Spark & Why?,” [Online]. Available: https://www.knowledgehut.com/blog/programming/scala-vs-python-vs-r-vs-java. [Acedido em Julho 2020].

[44] “Twitter4J,” [Online]. Available: http://twitter4j.org/en/.

[45] “ The OAuth 2.0 Authorization Framework,” [Online]. Available: https://tools.ietf.org/html/rfc6749.

[46] “JTwitter - the Java library for the Twitter API,” [Online]. Available: https://www.winterwell.com/software/jtwitter.php.

[47] “Regular-Expressions,” [Online]. Available: https://www.regular-expressions.info/.

[48] “Machine Learning Library (MLlib) Guide,” [Online]. Available: https://spark.apache.org/docs/latest/ml-guide.html. [Acedido em Junho 2020].

[49] “Spark SQL, DataFrames and Datasets Guide,” [Online]. Available: https://spark.apache.org/docs/latest/sql-programming-guide.html. [Acedido em Maio 2020].

[50] “TF-IDF,” [Online]. Available: https://spark.apache.org/docs/latest/ml-features.html#tf-idf.

[51] “Extracting, transforming and selecting features (StringIndexer),” [Online]. Available: https://spark.apache.org/docs/latest/ml-features.html#stringindexer. [Acedido em Junho 2020].

[52] A. Domingos, H. Ferreira, P. Rijo, C. Vaz e A. P. Francisco, “Degrees of separation on a dynamic social network.,” *In Mining and Learning with Graphs - Knowledge Discovery and Data Mining (MLG-KDD’13), ACM,* 2013.

[53] G. Tzinos, “GitHub - gtzinos/BigData-Graph-Analysis: Probably the first scalable and open source triangle count based on each edge, on scala and spark for every Big Dataset. (Louvain),” 2018. [Online]. Available: https://github.com/gtzinos/BigData-Graph-Analysis. [Acedido em Julho 2020].

[54] “Spring Web MVC,” [Online]. Available: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html.

[55] C. Hadjinikolis, “Labelled-Brexit-Tweets | Kaggle,” [Online]. Available: https://www.kaggle.com/chadjinik/labelledbrexittweets. [Acedido em Julho 2020].

[56] G. James, D. Witten, T. Hastie e R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer, 2013.