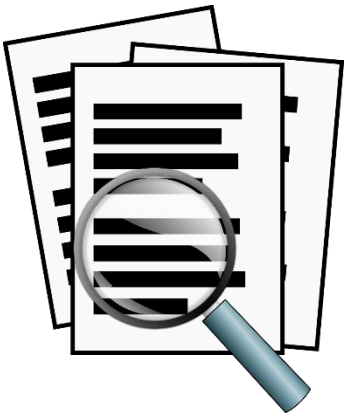


Ferramenta de exploração de anúncios públicos portugueses

Mariana Sofia Moreira da Silva Moreno
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Ciência de Computadores
2018

Orientador
Alípio Mário Guedes Jorge, Professor Associado, Faculdade de Ciências

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Agradecimentos

Em primeiro lugar quero agradecer ao meu orientador, o Professor Alípio Mário Guedes Jorge, por toda o apoio que me deu a definir os objetivos da dissertação, orientando-me nas diferentes tarefas e mostrando sempre disponibilidade e paciência para me ajudar. Quero agradecer de igual forma à minha família, em especial aos meus pais e irmã por toda a motivação que me deram ao longo desta fase transmitindo-me sempre confiança, ao meu namorado, José Sousa, por me apoiar em todos os momentos e às minhas melhores amigas, Mafalda Lemos e Marta Regina por me animarem em alturas de grande stress e ansiedade.

Resumo

A variedade e capacidade de armazenamento de dados tem crescido de forma paralela ao avanço tecnológico do *hardware* e *software*. Estima-se que 80% das informações armazenadas pelas empresas estão contidas em documentos de texto, sendo que o ideal seria organizar essas informações em bases de dados estruturadas ficando apenas com o essencial desses documentos, podendo utiliza-los de forma estratégica em processos de tomada de decisões[1]. Também o conteúdo *online* se encontra maioritariamente em formato textual sendo que a tendência é que, dia após dia, esse aumente expressivamente. A verdade é que com tanta informação disponível existe cada vez mais uma necessidade de filtrar essa informação retendo apenas o fundamental e é isso mesmo que o *text mining* pretende.
Neste trabalho serão usadas metodologias de *text mining* e de extração de informação para identificar elementos relevantes em anúncios públicos. Será montado um *pipeline* de extração para monitoração e leitura aumentada dos documentos.

Palavras-chave: Extração de Informação, Extração de Keywords, Análise Preditiva, Problemas de Regressão, Text mining, Data Mining, Anúncios Públicos.

Abstract

The variety and capacity of data storage has grown in parallel with the technological advancement of *hardware* and *software*. It is estimated that 80% of the information stored by companies is contained in text documents,the ideally to organize this information in structured databases with only the essential of these documents, and can use them strategically in processes of decision making [1]. Also the *online* content is mostly in textual format and the tendency is that, day after day, it increases expressively. The truth is that with so much information available there is more and more a need to filter this information by retaining only the fundamental and this is what *text mining* intends to do.
In this work, text mining and text extraction methodologies will be used to identify relevant elements in public announcements. An extraction pipeline will be mounted for monitoring and increased reading of the documents.

Key words: Information Extraction, Keywords extraction, Predictive Analytics, Regression problems, Text mining, Data Mining, Public Announcements.

Conteúdo

Lista de Tabelas

Lista de Figuras

Acrónimos

CCP Código dos Contratos públicos

TED Tenders Electronic Daily

RSLP Removedor de Sufixos da Língua Portuguesa

MSE Mean Squared Error

CPV Vocabulário Comum para os Contratos Públicos

CSS Cascading Style Sheets

DSL Domain Specific Language

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

IDE Integrated Development Environment

IDF Inverse Document Frequency

JSON JavaScript Object Notation

KEA Keyphrase Extraction Algorithm

KNN K-Nearest Neighbors

OCR Optical Character Recognition

RAKE Rapid Automatic Keyword Extraction

SVM Support Vector Machine

TF Term Frequency

TF-IDF Term Frequency–Inverse Document Frequency

URI Uniform Resource Identifier

URL Uniform Resource Locator

XML Extensible Markup Language

YAKE Yet Another Keyword Extractor

Capítulo 1

Introdução

1.1 Motivação

O acesso à informação por parte dos cidadãos é uma componente fundamental de qualquer democracia [2]. Assente em 4 pilares fundamentais - livre circulação de pessoas, bens/mercadorias, serviços e capitais - o Mercado Único (*Single Market*) é desde 1993 o acordo intergovernamental oficial instituído na Europa [3]. Por isso mesmo, os concursos públicos seguem o chamado Código dos Contratos Públicos [4], que estabelece as regras a aplicar à contratação pública e o regime substantivo dos contratos públicos que revistam a natureza de contrato administrativo. Além de seguirem este código, os concursos públicos que ultrapassem os valores contratuais específicos devem ser publicados no Suplemento do Jornal Oficial da União Europeia e devem ser divulgados em toda a União Europeia, contribuindo para uma maior transparência e assegurando a concorrência aberta e a gestão processual sólida. O acesso a este suplemento pode ser feito através do TED [5] (*Tenders Electronic Daily* - Diário Eletrónico dos Concursos), onde as informações gerais relativas a cada concurso são publicadas nas 24 línguas oficiais da União Europeia, assim como os anúncios das instituições europeias. No TED, são publicados mais de 1000 anúncios públicos diariamente.
Em Portugal existe uma outra plataforma que faz a publicação de todos os anúncios e contratos públicos portugueses, o BASE [6]. No entanto, e se tivermos em consideração que trimestralmente são publicados cerca de 2000 concursos públicos, há uma necessidade de facilitar a leitura desta quantidade enorme de dados disponibilizados. Posto isto, neste trabalho foi desenvolvida uma ferramenta, a *Cloudy*, que tem como intuito permitir uma leitura rápida e aumentada destes dados estimulando o cidadão comum a um maior conhecimento sobre os mesmos e promovendo uma maior transparência.

1.2 Objetivos

Os objetivos principais deste trabalho são:

1. Desenvolver um pipeline de extração de informação para obter anúncios públicos portugueses. Através do portal BASE [6], além dos anúncios públicos serão também extraídos meta-dados que lhes estão associados.
2. Proporcionar uma análise geral de um conjunto de anúncios, com gráficos e tabelas para uma leitura simples e rápida.
3. Extrair informações dos anúncios transferidos recorrendo a algoritmos de extração de *keywords/keyphrases*.
4. Permitir o estudo de anúncios públicos tendo como parâmetros os valores dos seus preços-base.
5. Explorar os temas de um conjunto de anúncios, que têm a mesma entidade emissora.
6. Oferecer aos utilizadores uma previsão de preços-base de anúncios, quando esta informação não se encontra disponível. Para isso serão estudados e posteriormente utilizados 3 algoritmos de previsão: *KNN*, *SVM* e *Random Forest* .

1.3 Estrutura da dissertação

- Capítulo 2 - Apresentação das várias tarefas de *text mining*, seguida de uma descrição de processos e algoritmos utilizados no desenvolvimento da ferramenta *Cloudy*.
- Capítulo 3 - Descrição das ferramentas usadas ao longo de todo o trabalho.
- Capítulo 4 - Casos de estudo - Caracterização dos diferentes utilizadores da *Cloudy* e tipos de dados extraídos do portal BASE [6].
- Capítulo 5 - Apresentação do desenho da arquitetura, de todas as etapas e de todos os painéis que compõem a *Cloudy*
- Capítulo 6 - Análise de resultados obtidos pela *Cloudy*, a um *corpus* de 5000 anúncios.
- Capítulo 7 - São descritas e fundamentadas as conclusões retiradas sobre todo o trabalho desenvolvido, assim como o possível desenvolvimento da ferramenta.

Capítulo 2

Revisão Do Estado da Arte

Text mining é uma subárea de *data mining* que utiliza algoritmos e ferramentas para analisar um conjunto de documentos de texto não estruturados tais como arquivos PDF, páginas Web, entre outros. Com *text mining* podemos agrupar documentos que tenham algo em comum, extrair, sumarizar e estruturar documentos ou mesmo interpretar a opinião de alguém por detrás de um texto[7].

2.1 Tarefas de *text mining*

- **Agrupar / organizar documentos (*clustering*)** - Divisão de dados em grupos de objetos semelhantes ou uma hierarquia que pode ser facilmente navegada pelo usuário.
- **Classificar documentos (categorização)** - Associação automática do documento a uma determinada classe, pertencente a um conjunto pré-definido de classes.

  Por vezes estes dois termos geram algumas confusões dada a dependência de um face ao outro. A categorização não é mais do que atribuição de nomes a cada um dos conjuntos de documentos agrupados de acordo com algum critério, ou seja, classifica cada um dos grupos identificados por *clustering*.
- **Recuperação de informação (*Information Retrieval*)** - Consiste na descoberta de documentos, dado um conjunto de dados, que melhor satisfazem a procura do utilizador
- **Extração de informação (*Information extraction*)** - Extração automática de informações em bases de dados, que são maioritariamente semi-estruturadas ou não estruturadas.
- **Análise preditiva** - Identificar se existe uma relação numa série de variáveis, que juntas conseguem determinar de forma satisfatória (na maioria dos casos) uma outra variável. Esta área tem sido bastante explorada principalmente na área da saúde, uma vez que é possível, por exemplo, usando dados de doentes e de um novo paciente, antecipar um diagnóstico do mesmo.

- **Sumarização de textos** - Por vezes é necessário um resumo de um texto ou um sumário de uma coleção com imensos documentos de forma a que o utilizador tenha uma ideia geral do tema que estes tratam. Para isso existem dois tipos de sumarização:
  - Sumarização extrativa - o sumário é inteiramente composto de palavras que estão contidas nos textos analisados;
  - Sumarização abstrata - o resumo contém informação sintetizada dos documentos com palavras que podem não estar presentes nos mesmos.
- **Análise de opiniões e sentimentos (*Opinion mining* e *Sentiment analysis***) - Atualmente as pessoas tendem a expressar a sua opinião *online* (sobre restaurantes, hotéis, filmes, produtos, etc.). Uma das tarefas mais exigentes de *text mining* passa por encontrar mecanismos para retirar conclusões destes textos, uma vez que, estas opiniões são um fator que influencia grande parte das pessoas.

Tal como referido nos objetivos, neste trabalho pretende-se desenvolver um *pipeline* de extração de informação, sendo assim importante o estudo prévio de algoritmos existentes neste setor. De seguida, são apresentados três algoritmos que realizam a extração de *keywords*.

2.2 Extração de *Keywords*

Keywords (ou *keyphrases*) [8] são palavras ou sequências de palavras que juntas definem o conteúdo do que está a ser analisado, sendo por isso a extração das mesmas bastante relevante quando se pretende analisar documentos ou fazer uma pesquisa nos mesmos. Um exemplo bastante comum são os documentos académicos que geralmente vêm acompanhados com um conjunto de *keywords* que definem o tema tratado. Neste relatório são apresentados três algoritmos que tratam desta extração:

2.2.1 RAKE - *Rapid Automatic Keywords Extraction*

Algoritmo que atribuindo um *score*, devolve todos os candidatos a *keyword* de um documento de texto [9]. Usa um ficheiro com *stopwords*, ou seja, palavras que nada revelam sobre o conteúdo dos documentos. Este extrator é o ideal para coleções dinâmicas, isto é, que tendem a aumentar ao longo do tempo uma vez que, não precisa de reler os documentos já analisados quando aparece um novo documento.

Etapas do algoritmo:

1. Divisão do texto pelos sinais de pontuação (","/"."/";"/-"/"?"/"!", etc);
2. Seleção das palavras ou conjunto de palavras candidatas a *keywords*, removendo as *stopwords* existentes do texto resultante do passo anterior:

3. Cálculo de propriedades - para cada uma das candidatas selecionadas no 2ºpasso são calculados o seu grau e frequência no texto. A frequência diz respeito ao número de vezes que a palavra ocorre. O grau de cada palavra é obtido somando a frequência ao número de vezes que a palavra aparece associada a outra, por exemplo, na Tabela 1 podemos observar que a palavra *information* aparece duas vezes (grau=2) mas como numa delas ocorre associada à palavra *stored* o seu grau aumenta ficando três.

4. São devolvidas as *keywords* com a sua respetiva pontuação.

Exemplo:

- **Texto:** *It is estimated that 80% of the information stored by the companies is contained in text documents, and the ideal would be to organize this information in structured databases, keeping only the essential of these documents, being able to use them in a strategic way in making processes of decisions.*
- **Resultado da remoção das stopwords no texto:** ['estimated', 'information stored', 'companies', 'contained','text documents', 'ideal','organize', 'information',structured databases', 'keeping',essential','documents', 'strategic',making processes','decisions']
- **Grau e Frequência de cada palavra:**

Palavra	Grau	Frequência	Grau/Frequência
structured	2	1	2
information	3	2	1.5
text	2	1	2
keeping	1	1	1
databases	2	1	2
stored	2	1	2
essential	1	1	1
ideal	1	1	1
decisions	1	1	1
companies	1	1	1
making	2	1	2
organize	1	1	1
process	2	1	2
contained	1	1	1
documents	3	2	1.5
strategic	1	1	1
estimated	1	1	1

Tabela 1: Candidatas a *keywords* com os seus repestivos atributos

- **Resultado do algoritmo:**

  [('making processes', 4.0), ('structured databases', 4.0), ('text documents', 3.5), ('information stored', 3.5), ('documents', 1.5), ('information', 1.5), ('strategic', 1.0), ('companies', 1.0), ('keeping', 1.0), ('estimated', 1.0), 'contained', 1.0), ('organize', 1.0), ('ideal', 1.0), ('decisions', 1.0), ('essential', 1.0)]

2.2.2 KEA - *Keyphrase Extraction Algorithm*

Algoritmo supervisionado, isto é, algoritmo que usa um conjunto de treino, cujas frases-chave já são conhecidas, para criar um modelo, usado depois para encontrar as *keyphrases* dos documentos pretendidos [10]. Tanto na conceção do modelo como na extração de palavras-chave, este algoritmo faz uma seleção de *keyphrases* que se divide em três passos:

1. **Tratamento do *input***

   - Os sinais de pontuação, parênteses e números marcam os limites das possíveis *keyphrases*;
   - Os apóstrofos são eliminados;
   - Palavras compostas são divididas.

2. **Identificação de frases-chave**

   Depois de concluído o primeiro passo, são consideradas frases-chave todas as frases que:

   - Cumprem um determinado tamanho definido pelo utilizador;
   - Não são formadas exclusivamente por nomes próprios;
   - Não começam ou terminam com uma *stopword*.

3. **Uso do método *Lovins***

   Com este algoritmo pretende-se reduzir o número de possíveis frases-chave a analisar, agrupando-as pelo seu *stem* (raiz da palavra)

Dado um conjunto de treino e executados estes passos, são atribuídas a cada *keyphrase* dois valores, o TFxIDF (*term frequency–inverse document frequency*) - que compara a frequência da *keyphrase* no documento com a frequência da mesma em toda a coleção de documentos (conjunto de treino + conjunto de teste) e a *first ocurrence* - que indica o número de palavras que precedem a primeira ocorrência da *keyphrase* no documento. A etapa seguinte consiste em comparar os resultados obtidos com as *keyphrases* dos respetivos autores, tendo em consideração as medidas anteriormente referidas, e construir um modelo em que é usada a técnica de Naïve Bayes. Com o modelo e uma coleção de documentos de texto, a seleção de candidatas a *keyphrases* é novamente realizada e posteriormente são entregues as *keyphrases* com maior *ranking*.

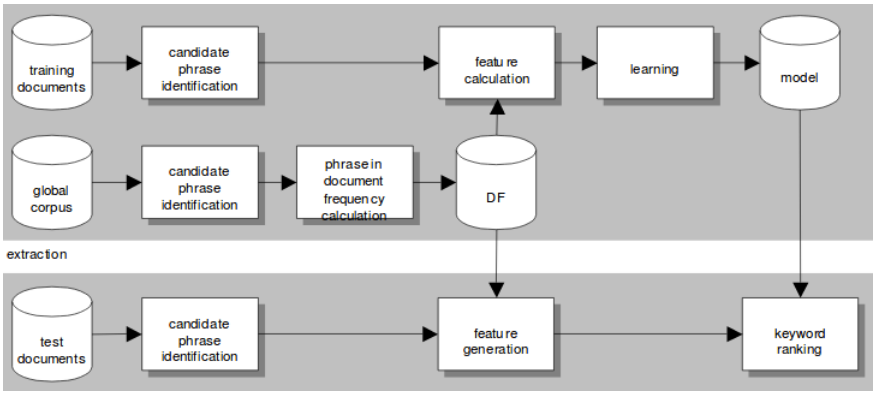


Figura 1: Imagem ilustrativa do algoritmo *KEA*
Fonte: KEA: Practical Automatic Keyphrase Extraction [10]

2.2.3 YAKE - *Yet Another Keyword Extractor*

Algoritmo com implementação em *python* que permite a extração de *keywords* de textos de diferentes tamanhos e em diferentes idiomas [11]. A principal vantagem face a outros algoritmos (como o algoritmo KEA 2.2.2) é que, como se trata se um algoritmo não supervisionado, é bastante útil quando o pretendido é a análise de pequenos textos. Este algoritmo é composto por 4 etapas: pré-processamento, extração de recursos, atribuição de um peso individual a cada termo, baseado nos *scores* anteriormente calculados, e geração de *keywords*.

1. **Pré - processamento** - divisão do texto por espaços ou caracteres especiais.
2. **Extração de recursos** - consiste no cálculo de 5 *scores* distintos referentes a cinco características do termo candidato a *keyword*.
   (a) ***Casing* (WCase)** - tem como objetivo valorizar acrónimos ou palavras cuja primeira letra se encontram em maiúscula e não estão no início de uma frase, sendo que quanto maior for o seu valor mais relevância a palavra terá;
   (b) ***Word Position* (WPosition)** - este valor é calculado através da média das posições da palavra no documento. Assim, palavras que ocorrem frequentemente no início do documento terão um peso menor do que aquelas que se encontrem no final do próprio, tornando-as mais importantes.
   (c) ***Word Frequency* (WFreq)** - procura valorizar palavras que aparecem frequentemente no texto, mas conjuga a frequência com a média de todas as frequências, de forma a penalizar *stopwords* como o "e", "a", etc.
   (d) ***Word relatedness to context* (WRel)** - tem em consideração o número de palavras diferentes que se encontram à esquerda e à direita do termo em questão, uma vez que, um termo tende a ser mais irrelevante se aparece constantemente perto de palavras diferentes, como o caso das *stopwords*.

(e) ***Word DifSentence* (WDifSentence)** - divisão entre o número de frases que contêm a palavra pelo número total de frases que contém o documento.

3. **Atribuição de um peso individual a cada termo** - depois de calculados os *scores*, o *YAKE* utiliza a seguinte fórmula:

$$S(w) = \frac{W_{Rel} * W_{Position}}{W_{Case} + \frac{W_{Freq}}{W_{Rel}} + \frac{W_{DifSentence}}{W_{Rel}}}$$

Para esta fórmula um valor baixo significa que a palavra é relevante no documento, ou seja, no caso de uma *keyword*, os *scores* como o *"Casing"*, a *"Word Frequency"* e o *"Word DifSentence"* devem ser altos ao contrário dos valores da *"Word Position"* e da *"Word relatedness to context"*.

4. **Geração de *keywords*** - este último passo do algoritmo atribui um *score* a todas os termos, que podem ser constituídos por uma, duas ou três palavras. Se o valor obtido for baixo, então o termo analisado é considerado uma *keyword*.

$$S(kw) = \frac{\prod_{w \in kw} S(w)}{TF(kw) * (1 + \sum_{w \in kw} S(w))}$$

No final, o algoritmo elimina candidatos semelhantes usando a medida *"Levenshtein distance"*.

Exemplo:

- **Texto:** *It is estimated that 80% of the information stored by the companies is contained in text documents, and the ideal would be to organize this information in structured databases, keeping only the essential of these documents, being able to use them in a strategic way in making processes of decisions.*
- **Resultado do algoritmo:** [(structured databases),(processes of decisions), (text documents),(contained in text), (making processes), (companies is contained), (documents), (information stored), (databases), (keeping), (decisions), (information in structured), (information), (structured), (text), (estimated), (organize this information), (processes), (contained), (organize)]

Além de algoritmos de extração de *keywords*, neste trabalho, são usados também algoritmos de aprendizagem para fazer uma previsão do preço-base de anúncios. No entanto, a aplicação destes algoritmos exige um pré-processamento dos textos. São descritas abaixo algumas das etapas primordiais desse pré-processamento.

2.3 *Stemming*

Stemming trata-se de um processo de reduzir uma palavra à sua raiz (*stem* em inglês). Este processo é bastante útil na área de recuperação de informação, principalmente em mecanismos de procura e pesquisas na internet, porque o mesmo tema pode ser procurado de várias formas diferentes e o uso deste processo proporciona melhores resultados, uma vez que as mesmas variantes de uma palavra têm uma só representação.

Para a língua portuguesa existe um algoritmo de supressão de sufixos, RSLP [12] que para a obtenção da *stem* de uma palavra:

- remove o sufixo, se a palavra em questão tiver um tamanho mínimo definido;
- acrescenta um outro sufixo em determinados casos;
- recorre a uma lista de palavras que não seguem os parâmetros gerais da nossa língua.

2.4 Representação de documentos

Tendo em consideração as tarefas que envolvem a área de data mining, torna-se evidente a necessidade de fazer comparação entre objetos, seja para conseguir agrupa-los e classifica-los ou para medir a semelhança entre eles. No caso destes objetos se tratarem de textos é essencial uma representação que permita tais tarefas.

2.4.1 Modelo de Espaço Vetorial

Neste modelo [13], cada documento é representado por um vetor de termos, tornando o espaço vetorial n-dimensional. Para calcular a proximidade entre dois vetores (documentos) calcula-se o cosseno do ângulo formado entre eles. Neste trabalho é explicada uma das medidas mais usadas na construção destes vetores, o TF-IDF.

2.4.1.1 TF-IDF

Medida estatística bastante utilizada na área do *text mining*, mais especificamente em recuperação de informação, uma vez que tem como intuito indicar a importância de uma palavra de um documento num corpus (conjunto de documentos). O valor TF-IDF[14] de uma determinada palavra aumenta de forma proporcional com o número de vezes em que esta ocorre no documento mas tem também em consideração a frequência da palavra em todo o corpus, o que ajudará a filtrar *stopwords*. Esta medida relaciona assim duas outras medidas:

- **TF (Frequência do termo)** - Divisão do número de vezes em que um termo aparece num documento pelo número de termos que esse documento contém.
- **IDF (Inverso da frequência nos documentos)** - Relaciona o número de documentos que constituem o corpus com o número de documentos em que o termo ocorre.
- **TF-IDF** - Traduz-se na multiplicação dos resultados dos dois termos anteriores.

2.5 Análise preditiva

A análise preditiva tem como objetivo usar uma grande quantidade de dados para compreender se existe uma associação sobre alguns desses dados, capaz de antever um outro dado. Esta análise pode ter duas finalidades:

1. Dado um conjunto de observações, antever o resultado de uma nova observação;
2. Usar um conjunto de observações para compreender os fatores que mais influenciaram a conclusão.

Modelo de previsão
Modelo que, usando um conjunto de observações, encontra o grupo de descritores que melhor determinam uma variável-alvo.

Existem dois tipos de problemas que um modelo de previsão pode abordar:

1. **Problema de classificação** - se a variável-alvo se tratar um uma variável nominal;
2. **Problema de regressão** - se a variável-alvo se tratar de uma variável numérica;

Sendo o conjunto de variáveis disponíveis grande e tendo como objetivo antecipar da melhor maneira um alvo, existem medidas que ajudam a entender se o modelo construído está a obter bons resultados.No caso de se tratar de um problema de classificação, pode usar-se o Error Rate. Esta medida calcula o número de casos em que a previsão estava errada. No caso de se tratar de um problema de regressão, utiliza-se a MSE (Mean Squared Error) que determina o desvio quadrado médio entre os valores reais e os valores obtidos com o modelo.

Uma vez que parte dos objetivos deste trabalho é oferecer um preço-base de um anúncio público, quando esta informação não se encontra disponível, ou seja, prever uma variável numérica, foram estudados três modelos:

2.5.1 K-Nearest Neighbors (KNN)

Modelo, em que dado um novo caso, procura as observações mais semelhantes e as usa para prever a variável-alvo que tanto pode ser numérica como nominal. Para medir a semelhança entre observações é usada a distância de Euclides ou a distância de *Minkowski*. Depois de calculadas as distâncias entre observações, a próxima etapa é determinar o número de vizinhos (k) que melhor

faz a previsão.

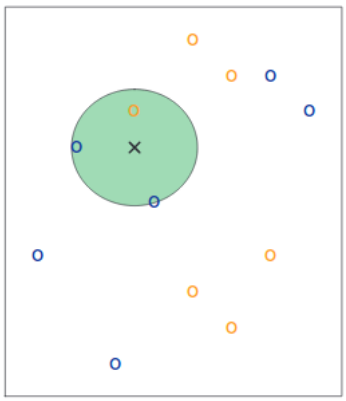


Figura 2: Representação do algoritmo KNN com k=3. Dada uma nova observação (x), são determinados os seus 3 vizinhos mais próximos.

Ao contrário da maioria dos modelos, o KNN não aprende com o conjunto de observações que tem, ele simplesmente compara-as com a nova observação para prever o valor da variável-alvo. Por esta razão, este modelo é bastante rápido na fase de treino, mas torna-se lento ao fazer a previsão, tendo que percorrer todas as observações e calcular a distância de todas face a que acaba de receber.

2.5.2 Support Vector Machine (SVM)

Este modelo de previsão consiste na criação de hiperplanos* capazes de dividir as amostras em classes distintas [15]. Inicialmente desenvolvido para lidar com problemas de classificação este modelo cria dois vetores de suporte que representam a maximização das margens entre o hiperplano e as amostras mais próximas de cada lado [16], tal como podemos ver na figura 3.

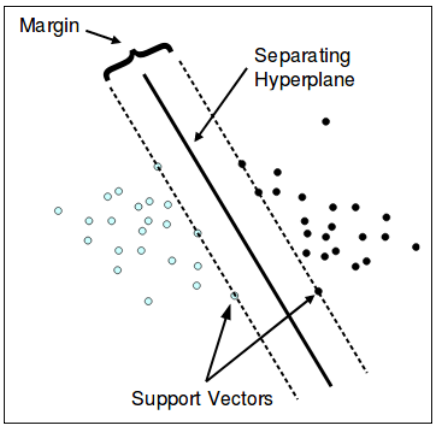


Figura 3: Separação ideal entre duas classes.
Fonte: Support Vector Machines - The Interface to libsvm in package e1071 [15]

Contudo, encontrar este hiperplano pode ser computacionalmente intenso quando as amostras

possuem várias dimensões e não estão distribuídas tão linearmente como na figura. Para tornar o cálculo deste hiperplano, menos complexo e custoso, é usado o *kernel trick* que consiste num conjunto de fórmulas matemáticas capazes de reduzir o número de dimensões das amostras.

Por vezes a margem entre o hiperplano e os seus vetores de suporte pode não se traduzir no melhor plano de divisão, e como o objetivo é maximizar as margens torna-se necessário aceitar determinados erros de forma a calcular uma melhor solução. Para isso é utilizado o parâmetro *Epsilon* para aumentar os limites dos vetores de suporte e *Cost* para dar uma penalização aos erros encontrados.

***** Num espaço p-dimensional, um hiperplano é um sub-espaço com dimensão p-1. Por exemplo, num espaço bidimensional, o hiperplano é uma linha.

2.5.3 Random Forest

Consiste na criação de árvores de decisão, escolhendo, para cada árvore, um conjunto aleatório de observações que podem conter observações repetidas (*bagging samples*). Assim para cada modelo que compõe o *Random Forest* são efetuados os seguintes passos:

1. É selecionado um conjunto aleatório de observações no início da árvore;
2. Para cada ramo são escolhidas aleatoriamente um conjunto de variáveis e de seguida o modelo faz a melhor separação destas.

Quando surge uma nova observação, o que o algoritmo faz é:

- determina o valor que mais se repetiu em todas as árvores de decisão e atribuí-lo à variável-alvo da nova observação, no caso de o problema ser de classificação;
- calcula a média dos resultados obtidos para a variável-alvo de cada uma das árvores e atribuir esse valor à variável-alvo da nova observação, no caso de se tratar de um problema de regressão.

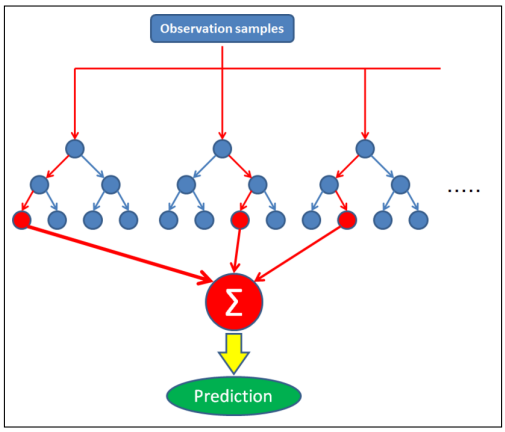


Figura 4: Exemplo da tomada de decisão do algoritmo Random Forest para um problema de Regressão.

Fonte: https://tjo-en.hatenablog.com/entry/2015/06/04/190000

Este algoritmo consegue bons resultados porque apesar de toda a aleatoriedade, junta o conhecimento de cada um dos modelos para tomar a decisão sobre a nova observação, fazendo com que os possíveis *outliers* não tenham tanta influência no resultado.

2.5.4 Performance Estimation

O objetivo da biblioteca *performance estimation* do R é obter uma estimativa confiável do erro de previsão de um determinado modelo. Com esta estimativa é possível não só, fazer análises e comparações entre modelos como determinar quais os mais adequados para um determinado conjunto de dados. Para calcular várias estimativas do erro do modelo é necessário dividir os dados em dois conjuntos, o conjunto de teste e o conjunto de treino. Assim o modelo será construído usando o conjunto de treino e fará as previsões das variáveis-alvo do conjunto de teste. Esta divisão de conjuntos pode ser feita de várias maneiras sendo que neste trabalho foi estudada e utilizada a técnica *cross validation*.

2.5.4.1 Cross-Validation

A ideia principal do *cross validation* é dividir um conjunto de dados em K partes iguais, usando uma das partes como teste e as restantes como treino para a criação do modelo, ou seja, quando o processo tiver terminado, são obtidos K modelos cada um com o seu erro associado. O erro final pode calcular-se como sendo a média de todos estes erros. Com esta técnica são obtidos vários erros, evitando problemas como a má escolha de uma amostra para os casos de teste.

Capítulo 3

Ferramentas de trabalho

Neste capítulo são apresentadas todas as ferramentas usadas no desenvolvimento da plataforma *Cloudy*.

3.1 *RStudio*

RStudio [17] é um *IDE* (Ambiente de desenvolvimento integrado) para a linguagem R, que foi usado neste trabalho para a escrita do código, instalação de *packages* necessários, acesso a variáveis e visualização de gráficos e *dataframes* (estruturas de dados específicas da linguagem R). Foi igualmente usado para a criação do *package Cloudy*.

3.2 *Bitbucket*

Repositório de código [18], escrito em *python*, que permite o controlo das várias versões de um projeto. Com o *bitbucket* é possível, entre outras coisas, a comparação de *branchs*, manter um histórico de *commits* e ter repositórios privados compartilhados. Este sistema de controlo de versões serviu neste trabalho, não só para implementar as várias *feautures* da *Cloudy*, como para ter *backups* para qualquer ponto no processo de desenvolvimento.

3.3 *Shiny*

Tendo como o objetivo, a criação de uma *app* onde a visualização dos dados é bastante fácil e interativa no *browser* foi utilizado o *package Shiny* [19]. Uma das grandes vantagens deste *package* é o facto do código da aplicação poder ser totalmente escrito na linguagem R, não sendo necessário quaisquer conhecimentos de outras linguagens como *HTML*,*CSS* ou mesmo *JavaScript*. Para o carregar basta:

```
library (shiny)
```

O primeiro passo para a utilização deste *package* é a criação de três funções: *app*, *ui* e *server*, sendo a primeira responsável por chamar as outras duas, a segunda respeitante à estrutura que a *app* terá, e a terceira controla as opções que o utilizador escolheu, as funções a chamar tendo em conta tais opções e qual o resultado a mostrar no final deste processo.

3.3.1 Dashboards

O pacote *shinydashboard* permite a construção fácil do painel que a *app* terá. Esse painel, definido na função *ui*, é composto por 3 partes (*header*, *sidebar* e *body*) e a sua estrutura base é obtida com o seguinte código:

```
library(shinydashboard)
dashboardPage(
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody()
)
```

3.3.2 Componentes

Usados para que haja comunicação entre o utilizador da *app* e a própria *app*. A tabela seguinte apresenta os *widgets* mais simples que o *package* disponibiliza:

Widget	**Descrição**
Button	Botão que quando clicado fará o *server* verificar certas ações que o utilizador procura
Date Input	Para que o utilizador defina uma data para a análise
Data Range	Possibilita ao utilizador escolher as datas, inicial e final
File Input	Usado para o utilizador selecionar um determinado ficheiro
Single Checkbox	Com este botão, o *server* pode verificar se o utilizador pretende usar a condição que lhe está associada
Radio buttons / Checkbox group	Conjunto de botões em que cada um deles representa uma alternativa disponível
Slider	Útil para que o utilizador defina de forma simples ou valores, mínimo e máximo, para cingir a sua procura
Text Input	Caixa de texto
Numeric Input	Espaço para a seleção de um número

Tabela 2: *Wigets* disponivéis no *package Shiny* do R

Para que o utilizador possa mostrar os seus resultados e análises de diversas formas, além

destes *widgets*, o *shiny* dispõe de algumas funções que convertem objetos *R* em diferentes *outputs* tais como:

Função	**Output**
dataTableOutput	Tabela
htmlOutput	raw HTML
imageOutput	Imagem
plotOutput	Gráfico
tableOutput	Tabela
textOutput	Texto
uiOutput	raw HTML
verbatimTextOutput	Texto

Tabela 3: Funções do *shiny*

3.3.3 Eventos

- *observeEvent* - permite que uma determinada tabela, gráfico, etc, apareça quando um determinado acontecimento se sucede;
- *eventReactive* - dado um acontecimento, altera um objeto R, tornando esse objeto reativo.

Por vezes, estes dois eventos criam confusões devido às suas semelhanças. A diferença entre eles, é que o primeiro evento muda o *output* sempre que um determinado parâmetro muda, enquanto o segundo só altera o *output* depois do acontecimento que fez com que o objeto R se tornasse reativo, voltar a repetir-se.

3.3.4 Alertas

Com o *shiny* é possível alertar o utilizador utilizando a função *showNotification*, que recebe como principais parâmetros: a mensagem que aparecerá, a duração do alerta e o tipo de alerta pretendido (mensagem, *warning* ou erro).
Para que esta mensagem apareça sob a forma de *popup* é necessário usar o *package shinyalert*.

Capítulo 4

Casos de estudo

Tal como referido em capítulos anteriores, a criação de uma ferramenta de trabalho que permite analisar anúncios será a ideia base do desenvolvimento deste projeto. No entanto, é necessário identificar primeiro quais os tipos de utilizadores e dados disponíveis para construir um projeto adequado, manuseável e que permita uma fácil interação pessoa-máquina.

4.1 Casos de uso

Antes de desenvolver a *app* é prioritário perceber que tipo de utilizadores terá interesse na mesma para que se consiga satisfazer a procura.
Como os documentos analisados se tratam de anúncios públicos, o primeiro tipo de utilizador, serão pessoas que pretendam uma leitura aumentada e rápida destes documentos, estando interessados em:

- gráficos/tabelas capazes de resumir o conjunto de documentos;
- gráficos em que sobressaiam os casos mais relevantes ou casos atípicos;
- dados sobre uma instituição em particular;
- os diferentes temas de anúncios face a diferentes preços-base.

Por outro lado, e tendo em conta que, para as análises feitas a estes documentos, são usadas técnicas de *text mining*, assim como algoritmos de extração de *keywords/keyphrases* e algoritmos de previsão, o segundo perfil de utilizador será alguém interessado em:

- perceber até que ponto a informação extraída consegue ser precisa face aos meta-dados fornecidos, isto é até que ponto o preenchimento das tabelas poderá ser feito automaticamente;
- estudar a diferença de resultados entre os algoritmos *RAKE* e *YAKE*;
- compreender em geral o funcionamento de um modelo de previsão.

4.2 Tipos de dados

No que diz respeito ao tipo de dados, estes serão extraídos do portal BASE [6]. Este *site* fornece informações atualizadas sobre contratos, anúncios e instituições públicas permitindo ao utilizador procurar por um destes temas em geral ou usar um conjunto de palavras para cingir a sua procura. Na figura 5 pode verificar-se que são devolvidas, por ordem decrescente do tempo, todos os anúncios relacionados com o tema pedido pelo utilizador e na figura 6 são devolvidas as entidades que correspondam à palavra-passe pedida com ligação a uma tabela que contém os gastos e ganhos dessas entidades assim como, os contratos por elas realizados.

Objeto do Contrato	Tipo de ato	Entidade	Preço base	Data de publicação	
Desenvolvimento Económico e Defesa do meio Ambiente -Mercados e Feiras - PARU - Plano Ação de Regeneração Urbana - Reabilitação do Mercado Municipal - 2ª Fase	Anúncio de procedimento	Município de Soure	715.755,91 €	03-08-2018	+
Aquisição de sanduiches preparadas, em ambiente natural, sem adição de atmosferas modificadas (Refª 01678/2018)	Anúncio de procedimento	Parques de Sintra - Monte da Lua, S. A.	199.900,00 €	27-07-2018	+

Figura 5: Resultado obtido de acordo com o parâmetro pedido (ambiente)
Fonte: http://www.base.gov.pt/Base/pt/

Descrição	País	NIF	
Associação para o Desenvolvimento da Faculdade de Ciências da Universidade do Porto	Portugal	504159313	+
Faculdade de Ciências da Nutrição e Alimentação da Universidade do Porto	Portugal	504060945	+

Figura 6: Procura de uma entidade (Faculdade de Ciências da Universidade do Porto)
Fonte: http://www.base.gov.pt/Base/pt/

É ainda possível obter algumas estatísticas sobre contratos públicos portugueses, nomeadamente dois gráficos, temporal e geográfico, que permitem perceber de forma rápida qual foi a evolução dos contratos públicos ao longo do tempo e quais são os distritos que mais dinheiro investem nestes contratos.

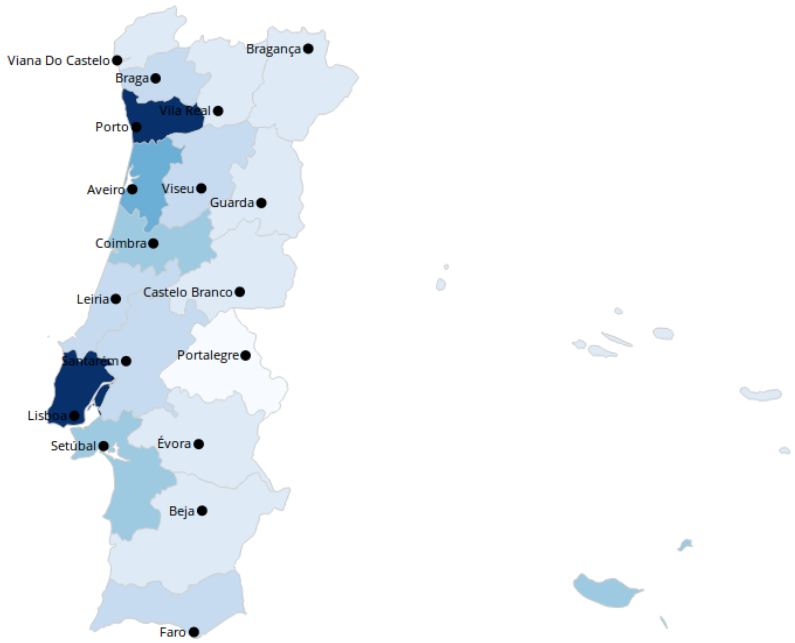


Figura 7: Gráfico geográfico que tem como métrica o valor dos contratos públicos portugueses Quanto maior a tonalidade do azul, maior o valor total dos contratos públicos efetuados nesse distrito.

Fonte: http://www.base.gov.pt/Base/pt/Estatisticas/GraficoGeografico

4.2.1 *Corpus*

Para este trabalho é necessário fazer *web scraping*, ou seja, extração seletiva de dados de uma página *web*, do *site BASE* [6] para obter os documentos que formarão o *corpus* usado no desenvolvimento da *app*. Além dos anúncios públicos são ainda extraídos deste site anúncios públicos portugueses, assim como os meta-dados que lhes estão associados. Os dados mais relevantes sobre um anúncio público são disponibilizados no BASE e, tal como será explicado no relatório, estes meta-dados podem ser extraídos automaticamente com informações contidas no próprio anúncio. Posto isto, todas as análises realizadas neste trabalho usarão estes dados e também a descrição sucinta de cada anúncio. Os meta-dados contêm:

- Número do Anúncio - é constituído por um conjunto de números e termina com a referência ao ano em que anúncio ficou acessível. Inicialmente parecia ser um identificador único para cada anúncio, mas depois de feitos mais *downloads* foi percetível que vários anúncios apresentam o mesmo Número do Anúncio;
- Diário da República;
- Entidade Emissora - entidade responsável pelo anúncio público;
- Descrição sucinta - resume o conteúdo do anúncio;
- Tipo de ato - indica o tipo de anúncio. que pode tratar-se de um anúncio de procedimento, anúncio de concurso urgente, aviso de prorrogação de prazo ou de uma declaração de retificação de anúncio;

- Tipo de Contrato - indica se o anúncio se trata de uma aquisição, locação, concessão, empreitada, etc;
- Modelo de Anúncio - especifica o tipo de concurso em que o anúncio se insere: concurso público, público urgente ou limitado por prévia qualificação;
- Preço-base - Preço indicado no anúncio;
- CPV (Vocabulário Comum para os Contratos Públicos) [20] - composto por um conjunto de números e uma descrição, procura cingir de forma rigorosa o tema do anúncio em questão;
- Prazo para apresentação de propostas - determina o limite de dias para a apresentação de propostas.

Neste capítulo foram identificados os casos de uso e os dados que serão explorados. O próximo capítulo contem a arquitetura e todas as etapas da *Cloudy* e ainda todas as funcionalidades e painéis que esta dispõe.

Capítulo 5

Plataforma Cloudy

5.1 Desenho da Arquitetura

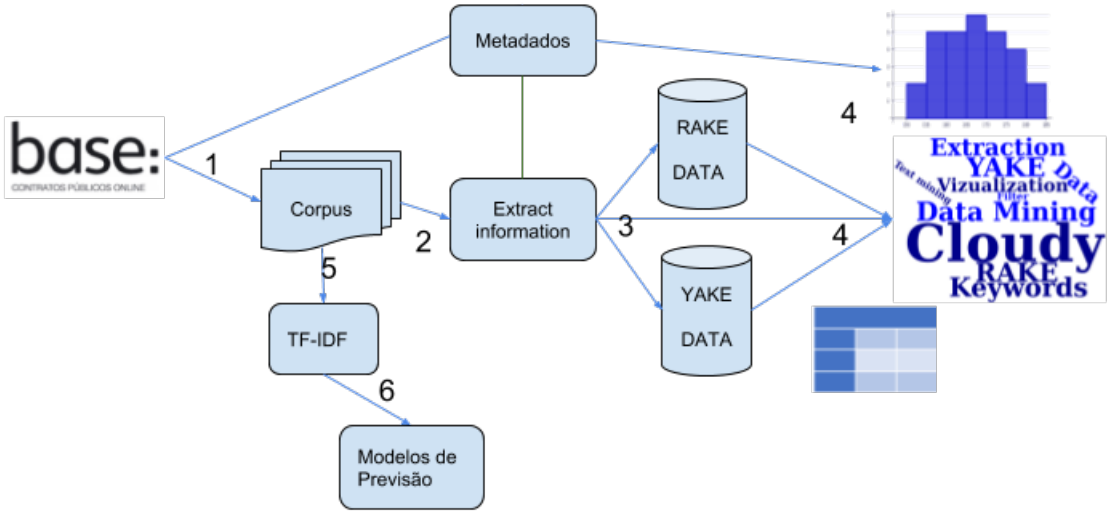


Figura 8: Cloudy

A figura acima apresenta as diferentes etapas que constarão na ferramenta.

1. *Web scraping* - definido o número de anúncios que o utilizador pretende, são feitos os *downloads* dos anúncios (que juntos formam o *corpus*) assim como os seus metadados;
2. Extração de Informação - são percorridos todos os ficheiros que compõem o *corpus*, de maneira a extrair dos mesmos os meta-dados que lhes estão associados;
3. Extração de *keywords/keyphrases* - Para cada anúncio são associados dois conjuntos de *keywords/keyphrases* (guardados em ficheiros .RData), o primeiro obtido usando o *YAKE* e o segundo usando o *RAKE*.
4. Filtragem e Visualização de dados - nesta etapa o utilizador poderá escolher o filtro pelo qual pretende analisar os anúncios (preço, instituição, *keyword*) e terá acesso a gráficos e tabelas que permitirão a leitura rápida dos anúncios.

5. TF-IDF
6. Modelos de Previsão - Atribuição de um preço-base a anúncios para os quais não existe um preço associado.

5.2 *Web Scraping*

Para a extração dos anúncios públicos (guardados em ficheiros *txt*), assim como os meta-dados que lhes estão associados (guardados no ficheiro *"advertisement.csv"*), o primeiro passo foi perceber como era feita a construção do *URL* para cada anúncio no *site*. Para tal, foi feita uma pesquisa geral dos anúncios, na página principal do BASE [6], que devolve todos os anúncios sendo o primeiro o mais recente. Através do *link* para esse primeiro anúncio, é extraído o seu *ID* e utilizando o *XPath* desse *link* é obtida a tabela do anúncio, que entre outras informações contém a ligação para o *pdf* do próprio. Depois, e decrementando este ID, são obtidos os restantes anúncios e respetivos dados até atingir o limite pedido pelo utilizador.

Com o web scraping são obtidos:

- O *corpus* formado pelos anúncios pedidos pelo utilizador;
- O ficheiro *"advertisement.csv"*, que contém os meta-dados desses anúncios.

Figura 9: Exemplo do *ID* de um anúncio

```
<a href="https://dre.pt/application/file/116344526" title="Ligação para o anúncio DRE"
```

Figura 10: Exemplo da ligação de um anúncio

5.3 Extração de Informação

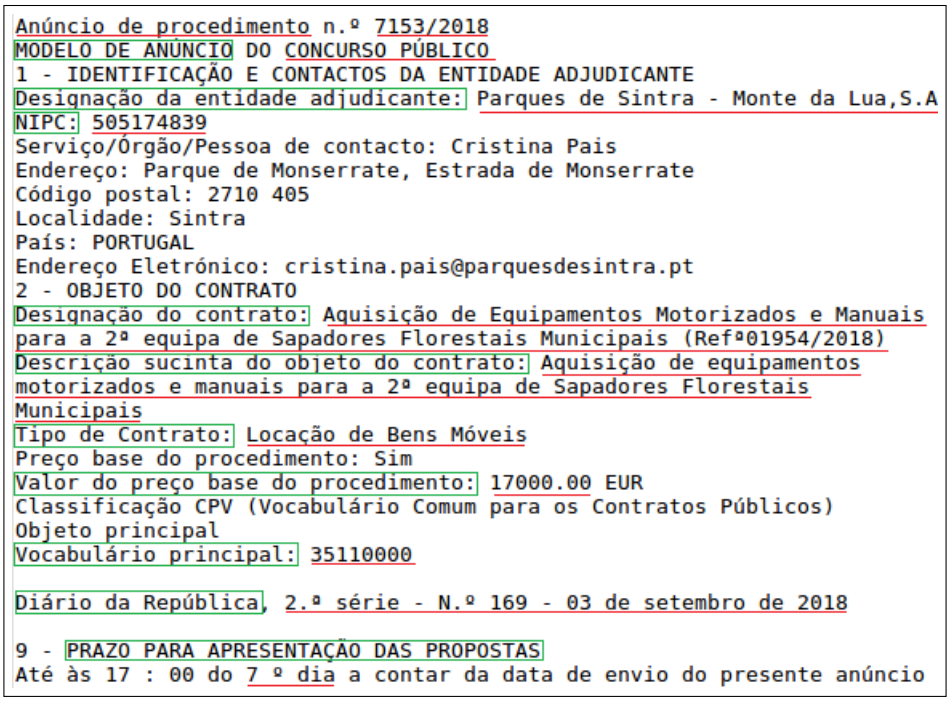


Figura 11: Exemplo de um anúncio público que segue a estrutura normal de um anúncio público português
. A verde estão a sequência de palavras a procurar;
. A vermelho estão os dados extraídos que irão constar na tabela *"extraction_information.csv"*.

Dado um anúncio, é verificado para cada linha do mesmo a ocorrência de uma determinada sequência de palavras. Na imagem 11 é percetível que, por exemplo, quando na linha aparece a sequência "Designação da Entidade Adjudicante:" à frente estará a entidade responsável pelo anúncio. Assim, são usadas todas as sequências necessárias para preencher a tabela *"extraction_information.csv"*. Desta forma é possível obter todos os dados pretendidos e que aparecem na tabela *"advertisement.csv"*, à exceção da coluna relativa ao CPV (Vocabulário Comum para os Contratos Públicos), uma vez que nos anúncios apenas se encontra o número do CPV, mas não a descrição deste. A esta tabela é acrescentada uma coluna "Descrição sucinta"que será usada para a extração de *keywords/keyphrases*.

⇒ Com a extração de informação é obtida uma tabela, que contém meta-dados de todos os anúncios do corpus, incluindo a coluna "Descrição Sucinta".

Tendo em conta que, o RAKE e o YAKE são os algoritmos mais indicados quando os textos a analisar são relativamente curtos (ver subsecção 2.2), este serão os extratores utilizados neste trabalho para a obtenção das *keywords* da coluna "Descrição Sucinta". Numa primeira fase, a *app* executava estes algoritmos a cada pesquisa efetuada pelo utilizador e de seguida devolvia as *keywords/keyphrases*, não guardando quaisquer informações relativas a esta pesquisa. No

entanto, percorrer a tabela a cada consulta aumentava consideravelmente o tempo de resposta. Assim sendo, a solução usada consiste em executar estes dois algoritmos para toda a tabela aquando dos *downloads* dos anúncios e guardar os dados em dois *dataframes*, que se tratam de uma estrutura de dados em que as linhas representam as diferentes observações e as colunas as diferentes variáveis, aos quais se acedem sempre que o utilizador realiza uma consulta. Nestes *dataframes*, as informações são guardadas por ordem crescente do preço-base de cada anúncio, tornando a filtragem por preços mais rápida. Uma vez que estes dados são usados frequentemente, para garantir que estes não se percam no encerrar de cada a sessão, os *dataframes* de cada extrator são guardados num ficheiro *RData* [21]. Além deste tipo de ficheiro possibilitar o acesso aos dados sempre que necessário, permite ainda apagar, modificar e acrescentar outros dados no futuro, o que é essencial neste trabalho, uma vez que o utilizador poderá fazer *downloads* de anúncios regularmente.

Para guardar os ficheiros *RData* basta:

```
save(YAKE_data,file="YAKE_data.RData")
save(RAKE_data,file="RAKE_data.RData")
save.image()
```

Sempre que se pretende aceder a um dos dataframes basta carregar o mesmo usando a função *load*.

```
load("YAKE_data.RData")
load("RAKE_data.RData")
```

ID	Numero do Anuncio	Preço Base	Keywords	Instituição	NIF	CPV
153626	6778/2018	1390000	c("habitação pública", "públ...	CMPH - DomusSocia...	505037700	45453000
98862	10653/2017	1391000	c("industriais", "comunicaç...	Águas do Douro e P...	514310774	30210000
152476	6441/2018	13945	c("lourosa", "comércio", "ru...	Município de Santa ...	501157280	45110000
154334	7162/2018	139518	c("verde", "alienação", "pel...	CARAM - Centro de ...	511259085	18620000
99527	29/2018	1395650	c("tecnologia led", "ilumina...	Electricidade dos Aç...	512012032	34928500
152495	6448/2018	139584	c("serviços médicos", "sant...	Unidade Local de Sa...	508094461	85100000
99086	306/2017	139725	c("all risks", "bens patrimon...	Hospital Professor D...	503035416	66510000

Figura 12: Imagem do *dataframe* que guarda as *keywords/keyphrases* das descrições sucintas, obtidas usando o extrator *YAKE*.

Dados um preço mínimo e máximo, o primeiro passo é percorrer o *dataframe "YAKE_data"* ou *"RAKE_data"*, dependendo do que o utilizador pedir, para obter as *keywords/keyphrases* dos anúncios cujos preços-base se encontram no intervalo pretendido.

ID	Numero do Anuncio	Preço Base	Keywords	Instituição	NIF	CPV
153626	6778/2018	1390000	c("habitação pública municipal...	CMPH - DomusSocial - Em...	505037700	45453000
98862	10653/2017	1391000	c("aquisição", "serviços", "com...	Águas do Douro e Paiva, S....	514310774	30210000
152476	6441/2018	13945	c("demolição", "comércio", "pa...	Município de Santa Maria ...	501157280	45110000
154334	7162/2018	139518	c("alienação", "verde", "peles",...	CARAM - Centro de Abate ...	511259085	18620000
99527	29/2018	1395650	c("iluminação pública equipada...	Electricidade dos Açores, S...	512012032	34928500
152495	6448/2018	139584	c("serviços médicos", "serviço"...	Unidade Local de Saúde d...	508094461	85100000
99086	306/2017	139725	c("responsabilidade civil geral"...	Hospital Professor Doutor ...	503035416	66510000

Figura 13: Imagem do *dataframe* que guarda as *keywords/keyphrases* das descrições sucintas, obtidas usando o extrator *RAKE*.

5.3.1 *Keywords*

Etapas:

1. É criado um corpus, em que são removidas todas as palavras que se encontram nas listas *"BASE_stopwords.csv"* e *"portuguese_verbs.csv"*.
   ⇒ Na lista *"BASE_stopwords.csv"*, encontram-se todas as stopwords relativas a anúncios públicos (ex: ”anúncio”,”classificação”,”termos”) e a lista *"portuguese_verbs.csv"*, que contém os verbos portugueses mais frequentemente usados (ex:”compartilhar”,”trabalhar”);
   ⇒ No caso do *YAKE*, em que poderá devolver termos constituídos por palavras como “de”,”o”, etc, é necessário aplicar aos seus resultados a lista “regular_stopwords.csv”, que contém todas as palavras portuguesas que, independentemente do tema tratado, são *stopwords*.
2. É feita uma matriz que descreve a frequência de cada palavra nas diferentes descrições sucintas;
3. Com o objetivo de obter uma só frequência por palavra, é feita a soma das várias frequências dessa mesma palavra em cada descrição sucinta;
4. À tabela obtida é aplicada um filtro; (ver subsecção 5.3.3)
5. É formada a *worcloud* com os termos mais frequentes do conjunto de descrições sucintas.

5.3.2 *Keyphrases*

Para disponibilizar *wordclouds* apenas com as *keyphrases* de um conjunto de anúncios públicos, são executados todos os passos anteriores, à exceção do último. As frequências de cada *keyword*, servirão para avaliar a importância de cada keyphrase, ou seja, quanto maior é a frequência de uma *keyword*, maior serão os scores de todas as keyphrases que contêm essa *keyword*.

Exemplo:

keyword	freq
águas	4
análise	3
tratamento	1

Dada a tabela acima e as *keyphrases* “tratamento de águas” e “análise de águas” os *scores* de cada *keyphrase* seriam:

- "tratamento de águas- **1** (pontuação inicial de cada keyphrase) + **4** (frequência da keyword “águas”) + **1** (frequência da keyword “tratamento”) = **6**
- "análise de águas- **1** + **4** (frequência da keywords “águas”) + **3** (frequência da keyword “análise”) = **8**

A função *filter* receberá neste caso uma tabela com as *keyphrases* e os respetivos *scores*.

5.3.3 Filtro

O filtro aplicado pretende retirar todos os termos que, apesar de serem considerados pelos extratores relevantes de uma determinada descrição, não têm uma importância considerável no conjunto total dos anúncios públicos analisados. Para isso, a função *filter*, remove todas os termos da tabela cuja frequência/*score* seja igual a 1. De seguida determina a média de frequências *scores*, e só os termos que contenham uma frequência *score* igual ou superior a esse número serão incluídos na *wordcloud*.

5.4 Análise Preditiva

Depois de explorar um conjunto amplo de anúncios torna-se evidente que muitos destes não apresentam um preço-base. Com o intuito de oferecer ao utilizador um preço-base para estes anúncios foram estudados e posteriormente implementados na *Cloudy*, modelos de previsão.
Uma vez que o atributo a utilizar para fazer a previsão de preços é a "Descrição Sucinta"de cada documento, o primeiro passo é quantificar a importância de cada palavra nessa descrição (através da medida TF-IDF). Com as palavras, que constituem as descrições, como descritores e o preço do anúncio como variável-alvo este problema trata-se de um problema de regressão.

5.4.1 Pré-processamento

Antes de criar este espaço vetorial é essencial fazer um pré-processamento das Descrições Sucintas, isto não irá permitir reduzir o tempo e complexidade do espaço vetorial como também irá permitir uma limpeza destes resultados. Este pré-processamento consiste em:

1. Excluir "Descrições Sucintas"que sejam iguais a "_", visto que estas não fornecem qualquer informação;
2. Fazer a remoção dos números;
3. Colocar todas as Descrições Sucintas em letra minúscula;
4. Remover todas as *stopwords* usando os três conjuntos anteriormente descritos 1;
5. Remover a pontuação;
6. Fazer *stemming* das palavras usando o algoritmo RSLP, através da biblioteca *ptstem* [22];

5.4.2 TF-IDF

Realizado o pré-processamento, é aplicado o algoritmo do TF-IDF, através da função *weightTfIdf* atribuindo assim, um valor para cada palavra em cada documento. Posto isto, neste momento, está criada a base de dados pretendida para formar os modelos.

5.4.3 Performance Estimation

Neste trabalho foram aplicados 3 modelos tendo como amostra 5000 documentos. Para fazer a avaliação destes foi utilizada a biblioteca *Performance Estimation*[23]. O primeiro passo foi descobrir para cada modelo quais os valores dos parâmetros com os quais eram obtidos melhores resultados. Posteriormente, foi feita uma comparação do melhor de cada modelo, de forma a perceber dos três, qual o modelo mais adequado a este caso de estudo. Em todas as avaliações será utilizado o método *cross-validation* com o número de divisões dos dados em 10 e para o erro será usado o *mean square error (MSE).*

5.4.3.1 KNN - K-Nearest Neighbors

Para o estudo deste modelo, fez-se uma variação do parâmetro correspondente ao número de vizinhos, usando os valores 3, 10, 50 e 100. Como mostra a figura 14, o valor do erro tende a diminuir à medida que o número de vizinhos aumenta, mantendo-se estável nos últimos testes.

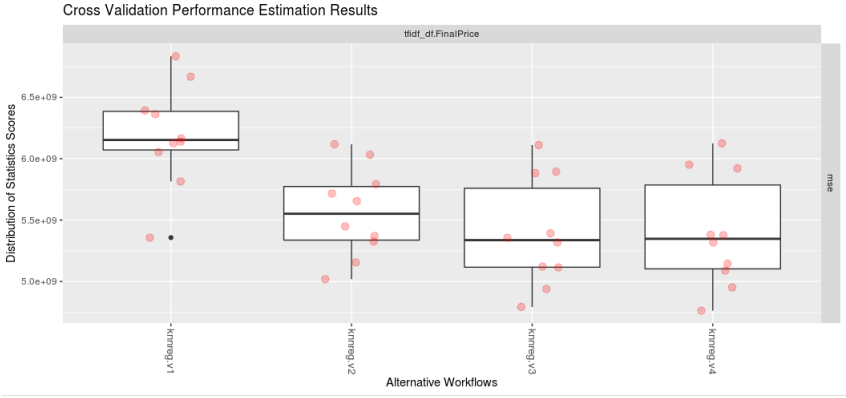


Figura 14: KNN com variação do parâmetro *neighbors*

5.4.3.2 SVM - Suport Vector Machines

Como neste modelo são vários os parâmetros a ter em conta, cada atributo foi analisado individualmente, começando pelo *cost*, que teve como valores 0.1, 1 e 10. tal como se pode constatar na figura 15 apesar de, não haver uma grande diferença entre os diferentes testes, o valor da média do erro foi mais baixo como o valor do parâmetro *cost* mais pequeno.

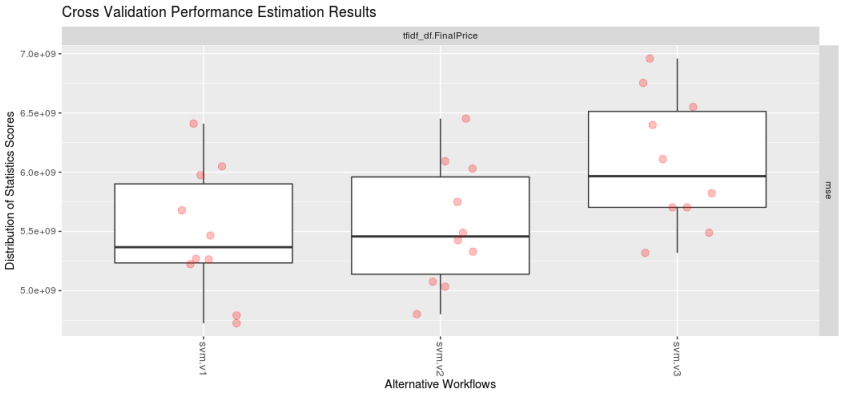


Figura 15: SVM com variação do *cost*

De seguida, fez variar o *epsilon*, concluindo-se que este parâmetro em nada influencia o resultado final, uma vez que para os diferentes valores atribuídos (0.01, 1 e 100), o resultado foi praticamente o mesmo.

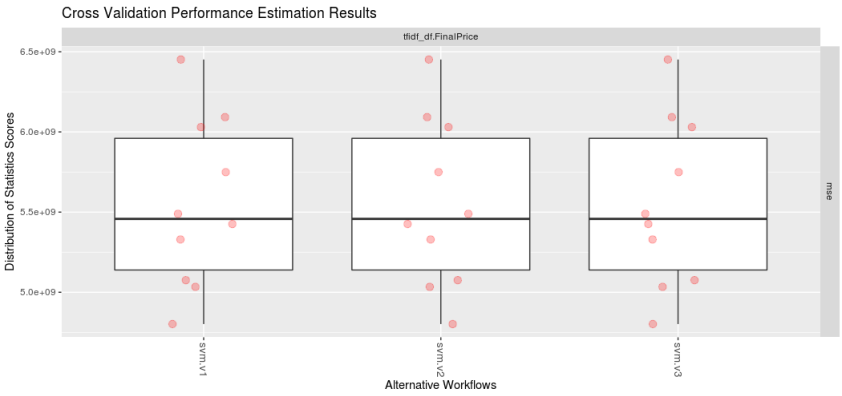


Figura 16: SVM com variação do *epsilon*

Por último foi feito um estudo do parâmetro *kernel*. Recorrendo à imagem 17 verifica-se que entre linear, sigmoide, polinomial e *radial bayesiano*, as funções *kernel* que conseguiram melhores resultados foram a linear e *radial bayesiano*.

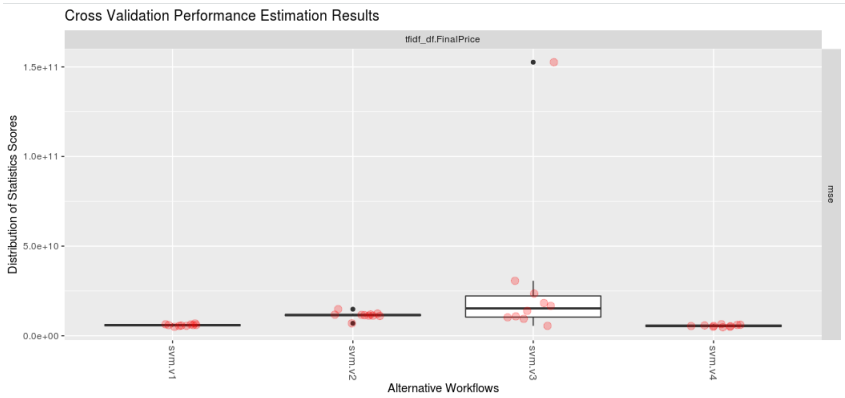


Figura 17: SVM com variação do *kernel*

5.4.3.3 Random Forest

O último modelo testado foi o *Random Forest* com variação do atributo que diz respeito ao número de árvores. Este modelo foi o mais custoso em termos de tempo, principalmente quando o valor do atributo ascendia os 500, contudo foi com estes valores mais altos que se verificou um valor médio do MSE mais baixo. Como podemos constar como a figura seguinte, os valores testados foram 10, 100, 500 e 1000.

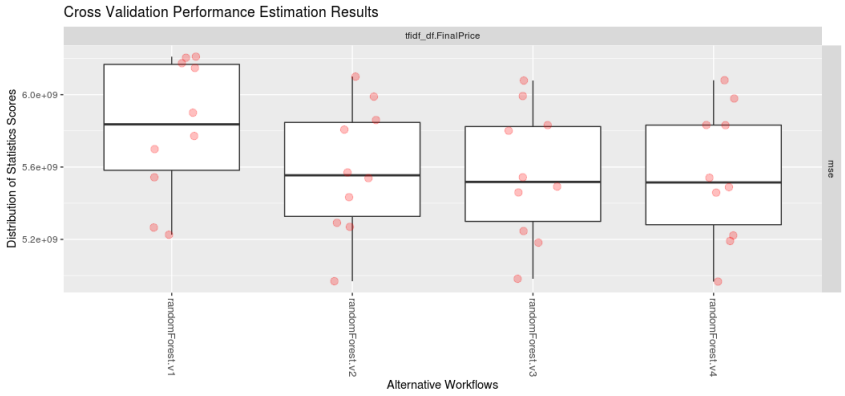


Figura 18: Random Forest com variação do parâmetro *ntree*

Analisando e comparando os modelos, com a melhor conjugação dos valores dos parâmetros, verifica-se que o KNN é dos três o que apresenta um menor erro, ainda que a diferença em relação aos outros dois seja quase impercetível. Em contra partida, o SVM é o que parece ter menor variância.19

Figura 19: Comparação dos três modelos estudados

Em suma, como o valor do *MSE* não apresentou grandes diferenças entre os modelos estudados, a *Cloudy* permitirá ao utilizador escolher o modelo que pretende, assim como os vários parâmetros de cada um deles.

5.5 Visualização

Quando o utilizador acede à *app* encontra o painel principal onde estão descritas as funcionalidades que esta dispõe. No lado esquerdo desse painel está uma *slide bar* com várias opções.

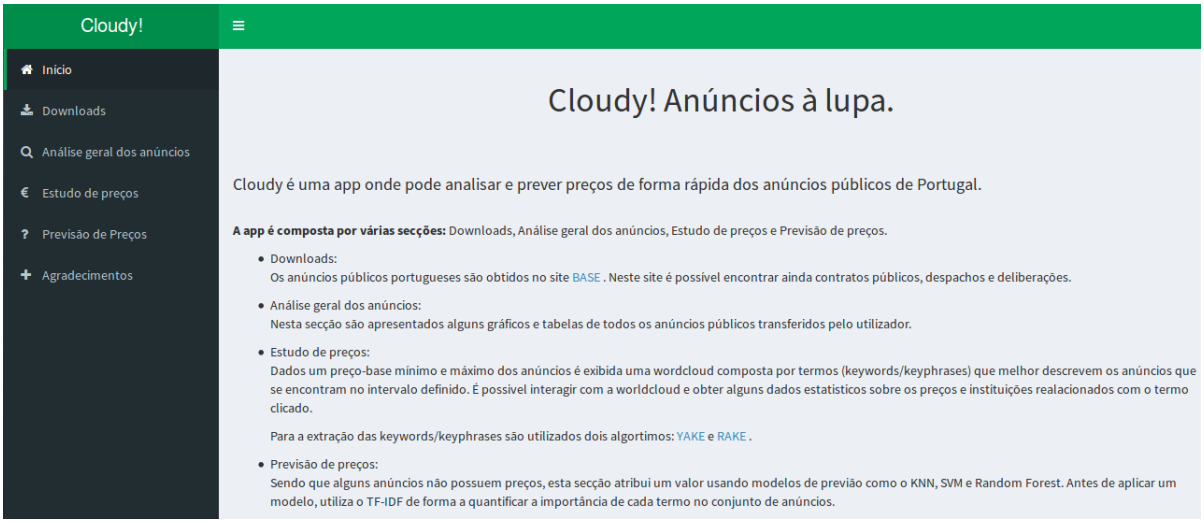


Figura 20: Painel inicial da *app*

Downloads

O utilizador pode transferir o número de anúncios públicos que pretende. Tal como já foi anteriormente dito (ver sub-secção 5.3), depois de transferidos os ficheiros, são extraídos os metadados associados a cada anúncio e determinadas as *keywords/keyphrases* das descrições sucintas usando tanto o extrator *YAKE* como o *RAKE*. Terminados os *downloads* e as extrações, o utilizador pode fazer de seguida as análises que pretende.

Figura 21: Downloads de anúncios públicos

Análise geral dos anúncios públicos

Nesta secção o utilizador pode encontrar gráficos e dados estáticos do conjunto de anúncios que tem. Logo no início do painel é exposta uma tabela sobre a precisão das informações usadas face aos meta-dados disponibilizados no BASE [6]. Esta tabela tem como intuito garantir ao utilizador a fiabilidade dos dados que serão analisados. De seguida é apresentado um gráfico de barras em que é possível escolher qual o atributo sobre o qual será feita a contagem de anúncios. Do lado direito desse gráfico está uma outra tabela, com a indicação dos números de anúncios que o utilizador tem no momento e quantos desses têm um preço-base definido. Para terminar, se o utilizador desejar saber a distribuição de preços-base de um determinado *CPV*, tem na divisão "Relação de Preço-CPV" a possibilidade de pesquisar o *CPV* pretendido no site *GOVIS* [20]. De seguida basta colocar o número de *CPV* e será mostrado um *box plot* com os diferentes preços-base dos anúncios que contêm esse *CPV*.

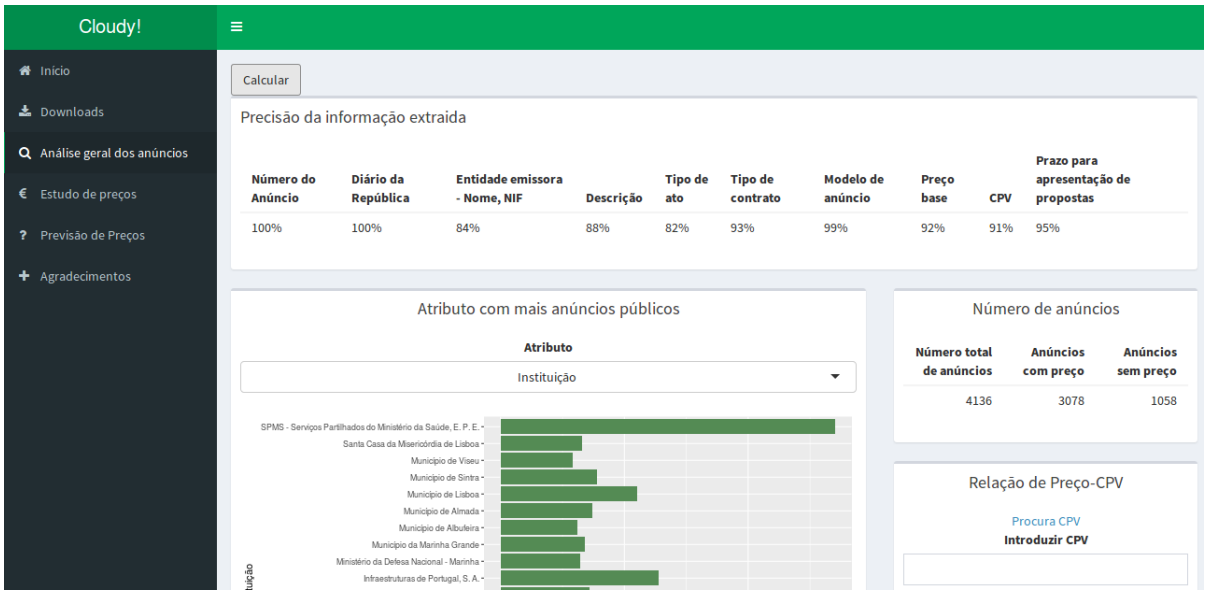


Figura 22: Análise geral dos anúncios públicos

Estudo de preços

Neste painel são pedidas as seguintes informações: o extrator, os valores mínimos e máximo do preço-base e ainda se o utilizador pretende obter uma análise dos anúncios através das *keywords* ou das *keyphrases* dos mesmos.

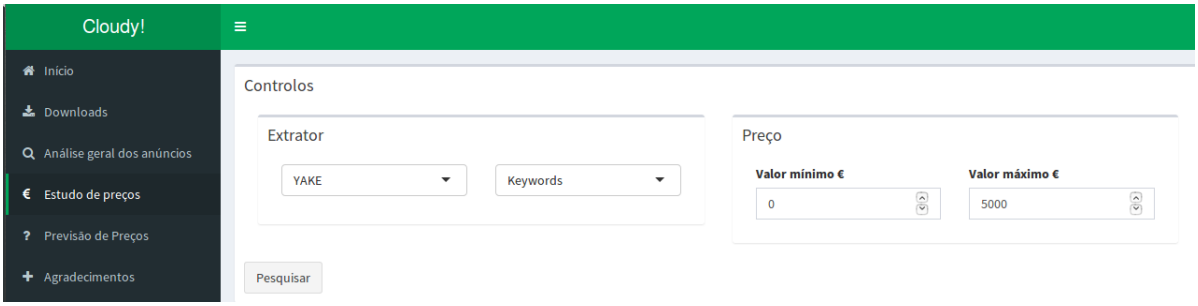


Figura 23: Painel Estudo de Preços

Determinados todos estes parâmetros, e depois de clicado o botão Pesquisar, será mostrada uma *wordcloud* com os termos mais frequentes.

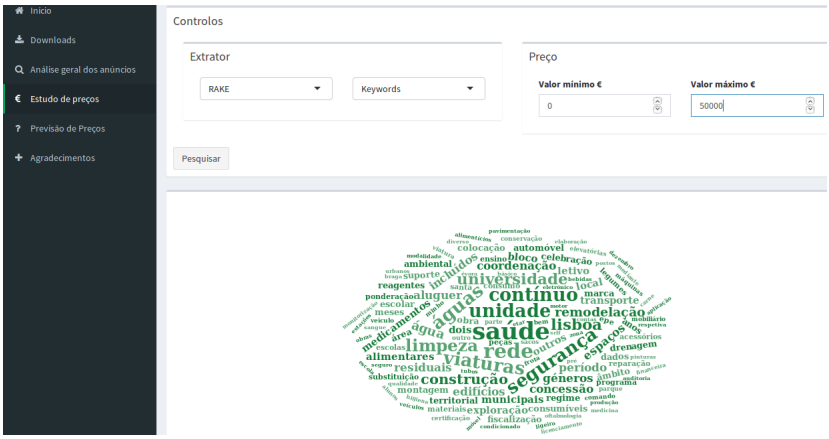


Figura 24: *Wordcloud* que segue os parâmetros determinados pelo utilizador

Ao percorrer a *wordcloud* apresentada, o utilizador poderá escolher a *keyword/keyphrase* sobre a qual pretende ter mais informações. Ao clicar, debaixo da *wordcloud* aparecerá o termo clicado acompanhado de duas opções: Histograma e Instituições. Ao clicar no botão Histograma surgirá um *popup* com um gráfico que apresenta a distribuição dos preços-base dos anúncios onde a *keyword/keyphrase* escolhida se encontra. Se o utilizador quiser, tem a opção de guardar o gráfico obtido, sendo apenas necessário clicar em *Download*.

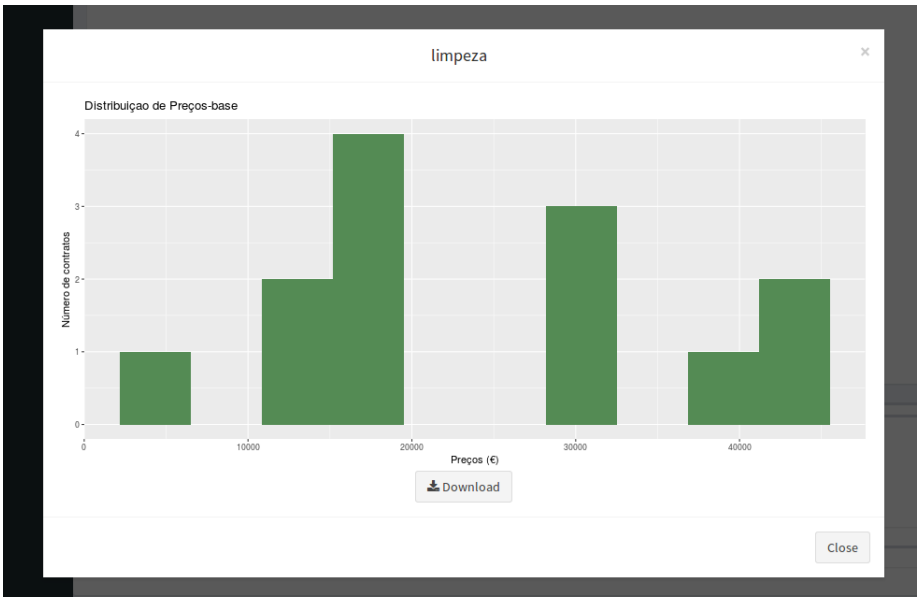


Figura 25: Distribuição dos preços-base dos anúncios onde a *keyword* "segurança"está presente

Ao clicar no botão Instituições surgirá uma tabela com todos os números dos anúncios e as instituições responsáveis pelos mesmos. Com esta tabela o utilizador tem a possibilidade de analisar melhor uma instituição em particular, bastando para isso clicar na instituição pretendida. Ao clicar, o processo é repetido, só que desta vez a *wordcloud* apresentada diz respeito à instituição clicada.

	Número do Anúncio	Instituição	NIF
1	5688/2018	Municipio da Marinha Grande	505776758
2	5294/2018	Municipio da Marinha Grande	505776758
3	205/2018	Serviços Intermunicipalizados de Águas e Resíduos dos Municípios de Loures e Odivelas	680009671
4	6513/2018	Municipio de Arcos de Valdevez	505211696
5	5237/2018	Escola Superior de Enfermagem de Coimbra	600081583
6	6512/2018	Municipio de Arcos de Valdevez	505211696
7	284/2018	Amadora Inovation, E. M., Unipessoal, L.da	504746383
8	6157/2018	Municipio de Vila Nova de Famalicão	506663264
9	6566/2018	Centro Hospitalar Tondela-Viseu, E. P. E.	509822940
10	11034/2017	Parques de Sintra - Monte da Lua, S. A.	505174839

Figura 26: Instituições responsáveis por anúncios onde a *keyword* "limpeza"está presente

Previsão de Preços

Neste painel, o utilizador terá que definir o modelo que pretende usar (cada um com os seus respetivos parâmetros) e qual o anúncio a analisar para obter um previsão do seu preço-base.

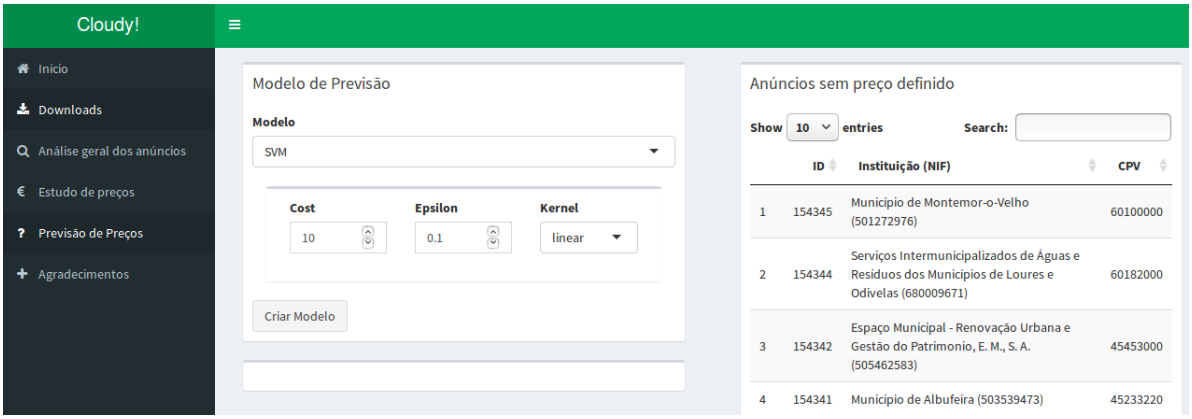


Figura 27: Painel Previsão de Preços

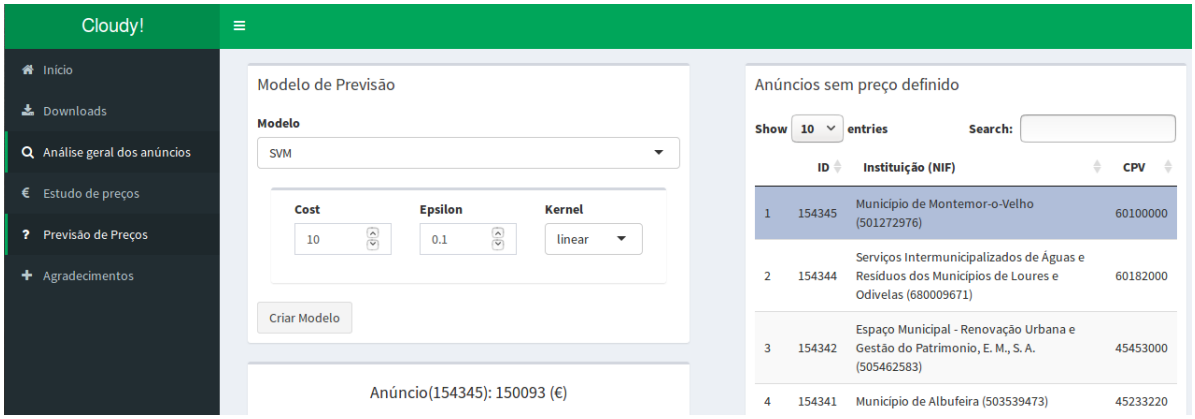


Figura 28: Previsão de um preço usando o modelo SVM

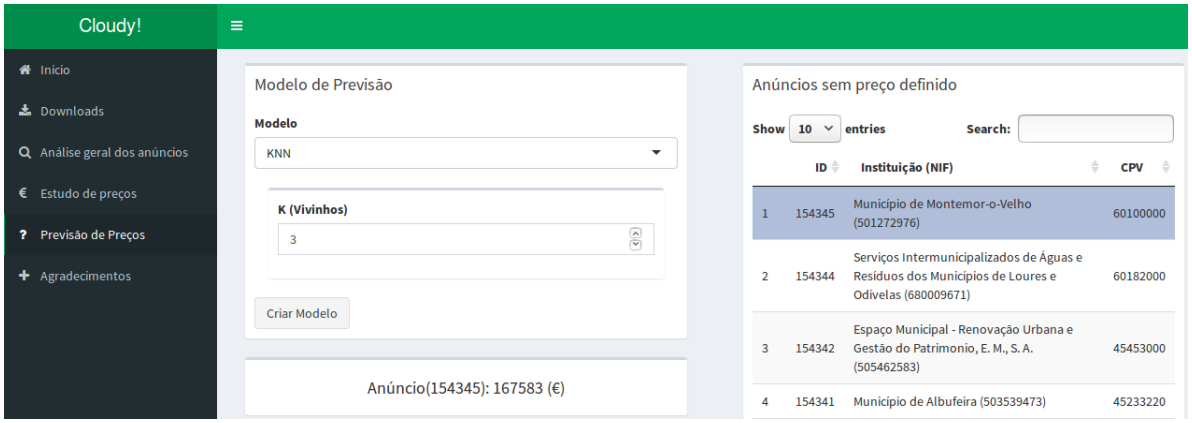


Figura 29: Previsão de um preço usando o modelo KNN

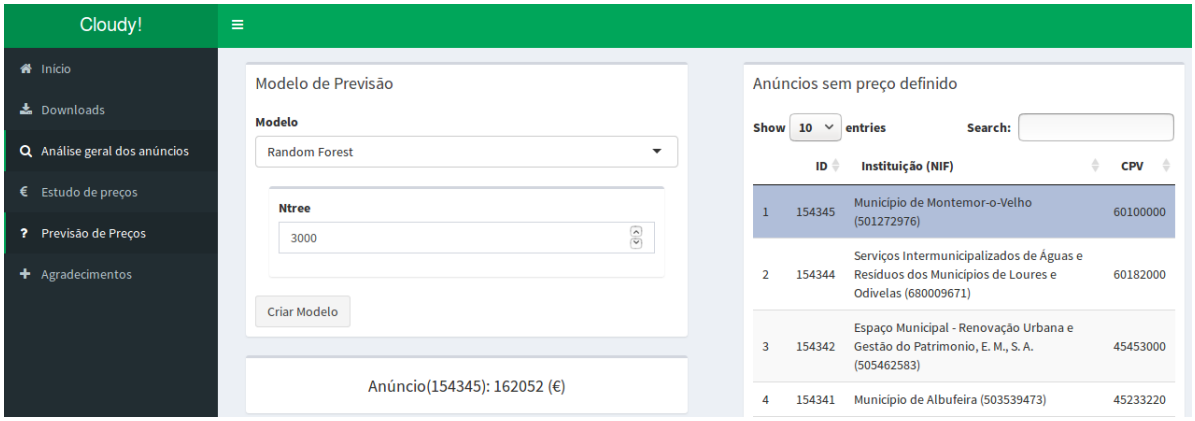
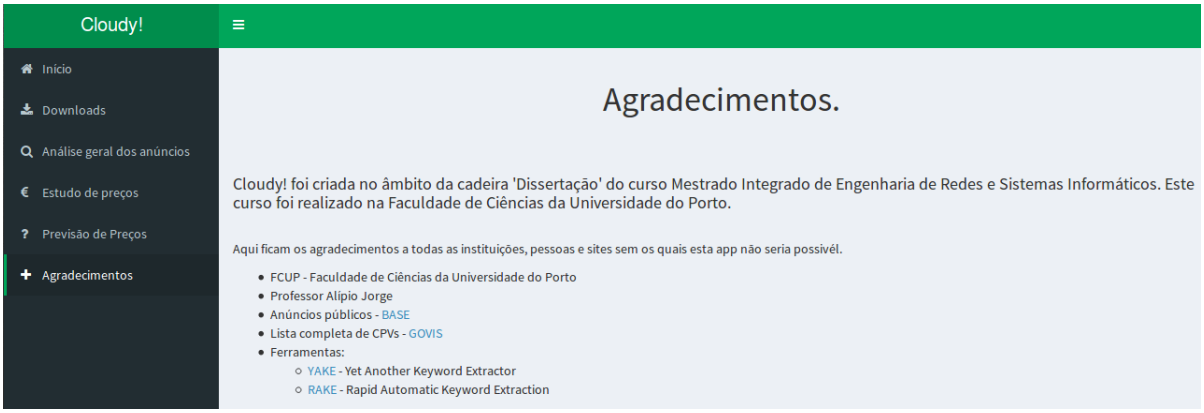


Figura 30: Previsão de um preço usando o modelo Random Forest

Agradecimentos

Painel de agradecimentos a todas as instituições, pessoas e sites sem as quais esta *app* não seria possível.



5.6 Construção da biblioteca *Cloudy*

Esta biblioteca pode ser acedida através do seu repositório no *bitbucket*. Depois de importar a biblioteca existe a função *cloudy()* que permite ao utilizador abrir e usar a aplicação no *browser*. Para instalar esta biblioteca:

⇒ install_bitbucket("MarianaMoreno/cloudy", ref="master")

Para a construção de uma biblioteca, o *RStudio* permite a criação dos vários ficheiros necessários, nomeadamente:

- DESCRIPTION - ficheiro com a descrição, versão, autor, licença, entre outros atributos do *package*;
- man - diretório com a documentação de todas as funções da aplicação;
- R - diretório onde estarão todas as funções e ficheiros.

Capítulo 6

Análise de Resultados

Nesta secção é feito um estudo consoante os resultados que a ferramenta mostra, tendo como base um *corpus* de 5000 anúncios.
Em primeiro lugar pode verificar-se que depois de realizados os *downloads*, o algoritmo *YAKE* comparativamente ao algoritmo *RAKE* demorou mais a fazer a extração de *keywords/keyphrases*. Tal pode justificar-se pelo facto do YAKE ser um algoritmo que requer mais cálculos, com maior complexidade de fórmulas, enquanto o *RAKE* utiliza uma lista de *stopwords* e realiza apenas o cálculo do grau e frequência das palavras.
Com os anúncios extraídos e os *dataframes* (YAKE_data.RData e RAKE_data.RData), o próximo passo é averiguar a qualidade dos dados que serão analisados. Posto isto, é consultada a tabela Precisão da informação extraída.

Precisão da informação extraida

Número do Anúncio	Diário da República	Entidade emissora - Nome, NIF	Descrição	Tipo de ato	Tipo de contrato	Modelo de anúncio	Preço base	CPV	Prazo para apresentação de propostas
100%	100%	81%	88%	85%	93%	99%	92%	91%	96%

Figura 31: Tabela Precisão da informação extraída para um corpus de 5000 anúncios.

Avaliando a tabela acima apresentada pode constatar-se que todas as percentagens calculadas se encontram acima dos 80%. Porém, o valor referente à Entidade Emissora (81%) é o que mais destaca. Este valor, que à partida parece escasso não representa um grande problema. Isto porque, em muitos casos a entidade obtida e a entidade apresentada nos meta-dados dos anúncios é a mesma, apenas difere numa palavra. Um exemplo de anúncios que contribuem para a diminuição desta percentagem é o caso em que nos anúncios, a entidade responsável é um determinado município, mas na tabela com os meta-dados dos mesmos, a entidade apenas contém o nome do município. Por exemplo: A entidade apresentada no anúncio é o "Município de Braga"e na tabela a entidade ser apenas "Braga". O mesmo acontece com os outros valores conseguidos nos atributos Tipo de ato, Tipo de contrato e Modelo de anúncio. Em relação aos atributos Descrição, Preço base, CPV e Prazo para apresentação de propostas, a percentagem

de precisão não é total porque em alguns anúncios esta informação não está disponível, mas na tabela do BASE, estes dados são divulgados.

Analisada a qualidade dos dados obtidos, é importante ter uma visão geral dos mesmos.

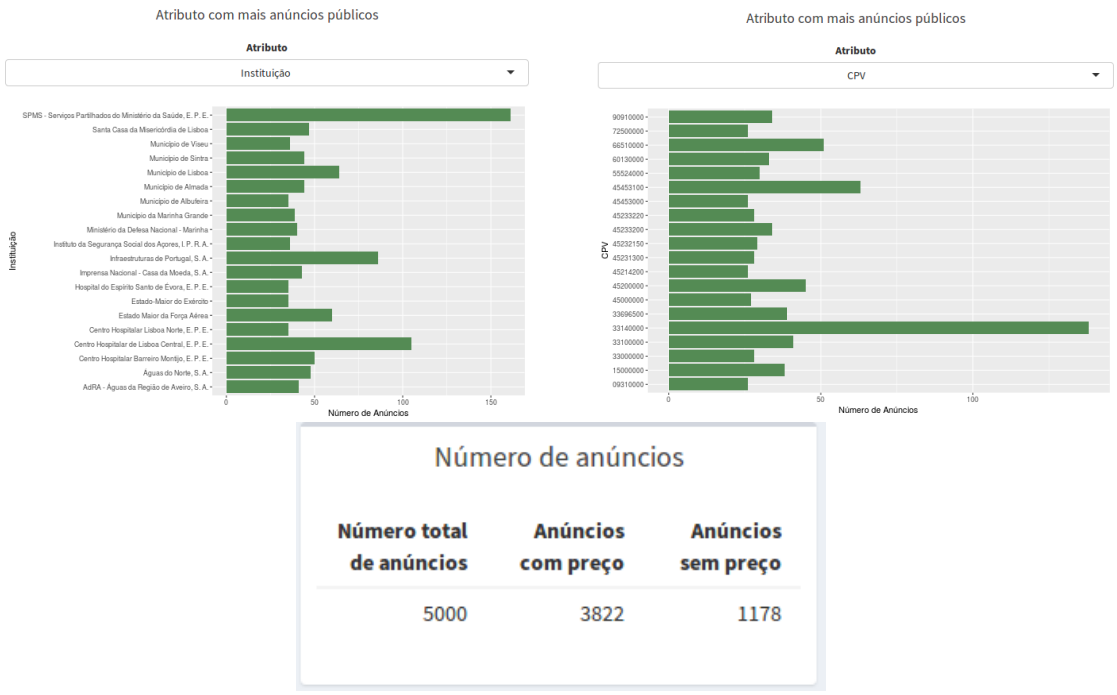


Figura 32: Distribuição dos anúncios pelo atributo selecionado / Número de Anúncios

De facto, poder procurar os anúncios partindo de um atributo em específico, faz com que o utilizador consiga não só uma visão do comportamento geral de cada atributo, mas consiga também perceber se há alguma ocorrência que se destaque de todas as outras. Nas imagens acima, é evidente que a instituição "SPMS - Serviços Partilhados do Ministério da Saúde, E.P.E"é a instituição que mais anúncios tem neste conjunto de documentos e que uma grande parte dos anúncios têm como CPV o número 33140000 (Material médico). Se o utilizador quiser a distribuição de preços dos anúncios com este CPV, pode utilizar o espaço Relação de Preço-CPV. Com o *boxplot* devolvido, pode observar-se que os valores destes anúncios rondam os 200.000 €, havendo 7 casos em que o preço-base é bastante superior aos preços dos restantes anúncios.

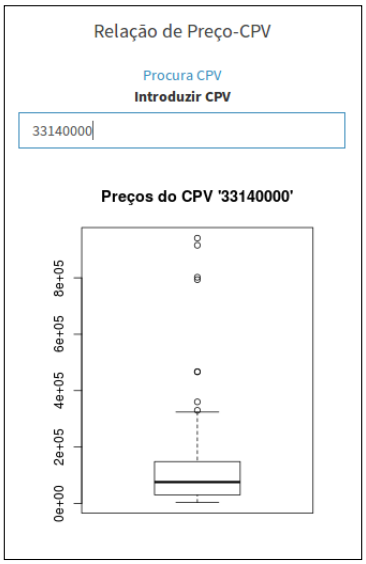


Figura 33: *Boxplot* relação Preço-CPV com CPV=33140000

Tendo agora, o objetivo de estudar os anúncios cujo preço se encontrem num determinado intervalo, define-se como ideia inicial, obter as *keywords* sendo o preço mínimo 0 e o máximo 100.000€.

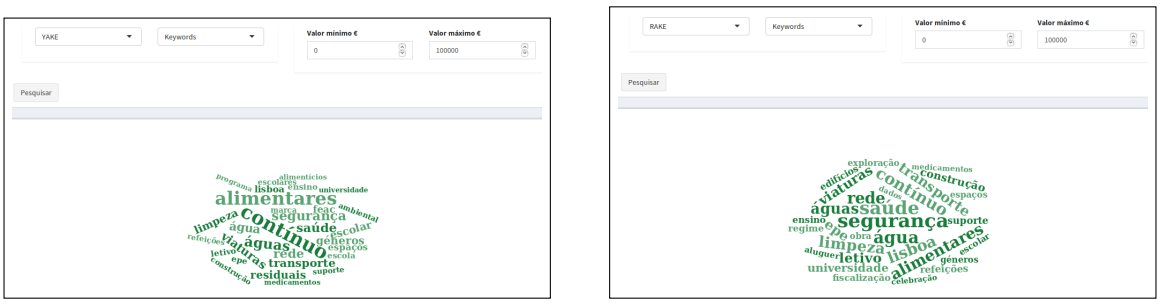


Figura 34: *Keywords* dos anúncios públicos cujo preço-base se encontra no intervalo definido

Com as *worclouds* obtidas, podem à partida tirar-se duas observações:

1. Apesar de serem diferentes, as *wordclouds* têm, na sua maioria, as mesmas palavras,que juntas permitem concluir os temas tratados nos anúncios que se encontram neste intervalo: Segurança, Águas, Saúde, Escola, Transporte, Alimentação, etc;
2. Os scores da *wordcloud* respeitante ao *YAKE* apresentam valores mais altos do que os da *wordcloud* do *RAKE*.

A primeira observação assenta no pressuposto que, apesar de se tratarem de algoritmos diferentes, quando o *corpus* sobre o qual estão a atuar é relativamente grande, os resultados devolvidos não serão tão diferentes; A segunda observação é a confirmação de que o algoritmo *RAKE*, a cada documento, devolve uma só vez uma determinada palavra/frase. Por sua vez, o *YAKE* pode devolver a mesma palavra, inserida em diferentes *keyphrases*, daí o *score* de cada *keyword* ser, na grande maioria, maior.

Para o mesmo estudo, mas desta vez para obter as *keyphrases* destes anúncios, as *wordclouds* resultantes são:

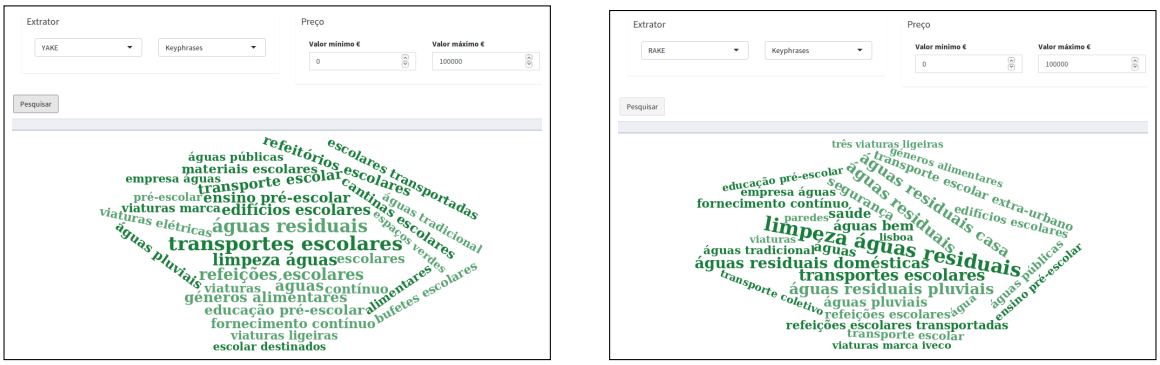


Figura 35: *Keyphrases* dos anúncios públicos cujo preço-base se encontra no intervalo definido

A primeira conclusão retirada é que o tempo de resposta se torna significativamente maior quando se pretende extrair *keyphrases* dos anúncios públicos. Contudo, quando analisados os resultados podemos depreender que o utilizador ganha informação face às *worclouds* respeitantes às *keywords*. Por exemplo, com estas *wordclouds* consegue perceber-se que os temas Escola e Transporte estão relacionados, assim como Escola e Alimentação. É também importante referir que o RAKE devolveu algumas *keyphrases* que não estão contidas na *wordcloud* do *YAKE*, mas quando estudados essas *keyphrases*, conclui-se que o número de anúncios em que estas se encontram são poucos sendo que estas *keyphrases* não são uma mais-valia no conjunto total de anúncios analisados.

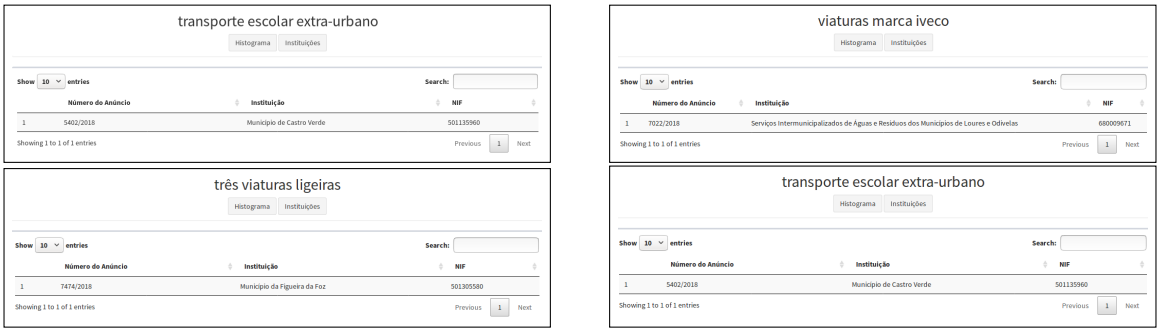


Figura 36: Informações sobre os vários anúncios que contêm as *keyphrases* devolvidas apenas pelo algoritmo RAKE.

Para ter uma ideia geral do comportamento do preço-base dos anúncios sendo dada uma palavra, foi feito o estudo de quatro *keywords* aleatórias (águas, limpeza, saúde e viaturas) no intervalo de preços anteriormente definidos (mínimo = 0€ e máximo = 100.000€). Com os histogramas, o utilizador conhece a distribuição de preços assim como o número de anúncios que contem o mesmo preço, tendo como parâmetro uma palavra.

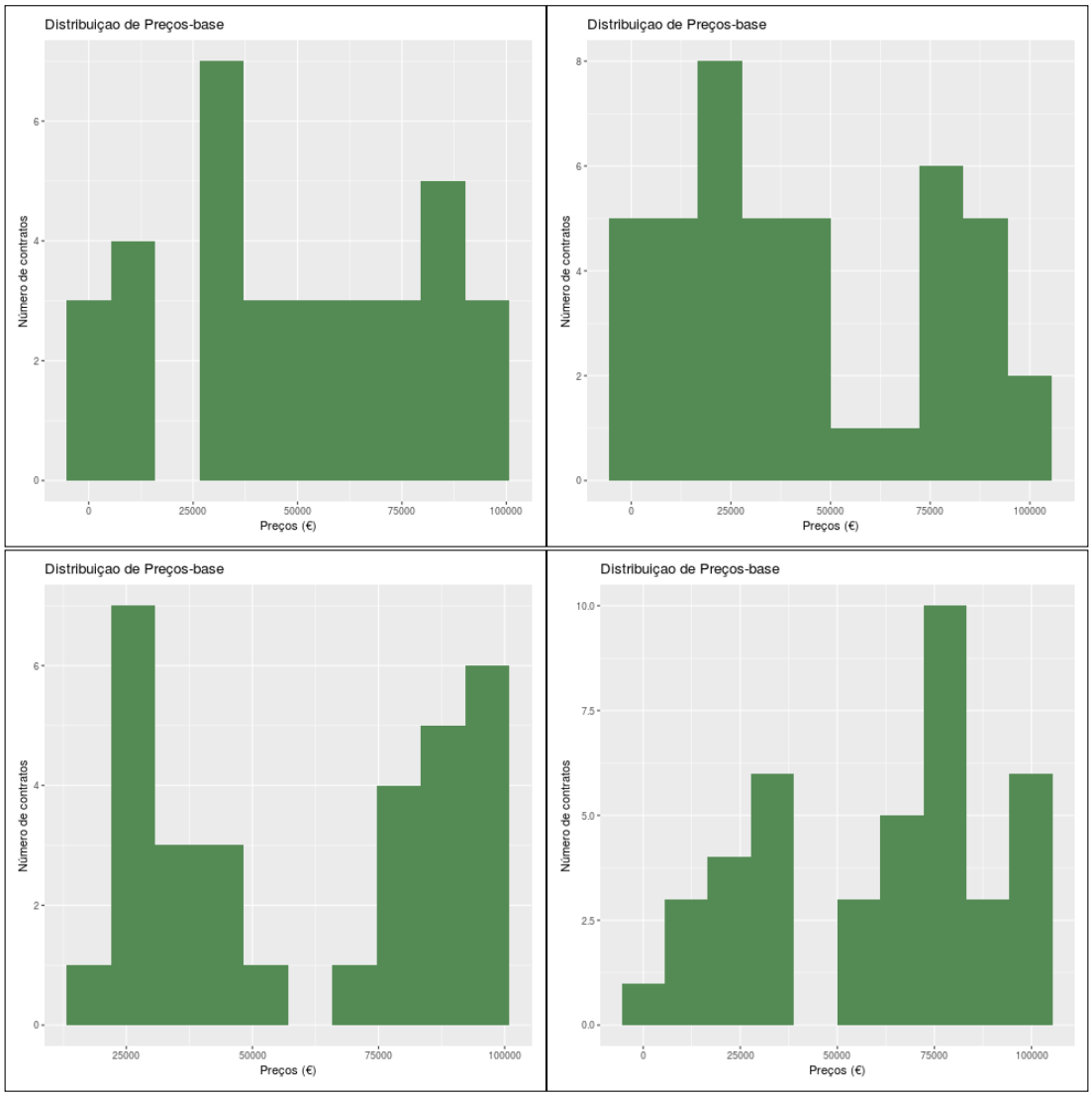


Figura 37: Distribuição dos preços-base tendo como base uma *keyword*

Como mostra a figura 37 o número de anúncios em que o preço-base está definido como 0€ é pequeno, tendo em conta o número total de anúncios. É visível também que nenhuma das palavras estudadas apresenta uma distribuição de preços regular, isto é, os anúncios que contêm essa palavra podem ter um preço desde o valor mínimo até ao máximo, não havendo um valor que se destaque expressivamente.

Com interesse em perceber o conteúdo dos anúncios que têm como responsável a instituição com o maior número de anúncios (verificar imagem 32) neste conjunto foi pesquisada a instituição e obtidas a seguintes *wordclouds*.

Figura 38: *Wordclouds* com as *keywords* de uma instituição
À direita a *wordcloud* obtida com o algoritmo *YAKE* e à esquerda com o algoritmo *RAKE*

Figura 39: *Wordclouds* com as *keyphrases* de uma instituição
À direita a *wordcloud* obtida com o algoritmo YAKE e à esquerda com o algoritmo *RAKE*

Mais uma vez é visível, com as imagens, que apesar de, as *wordclouds* que contêm as *keyphrases* demorarem mais a ser formadas, devolvem mais informação ao utilizador. Tal como seria de esperar, as aquisições e matérias abordadas nos anúncios da instituição pesquisada, Instituição: SPMS - Serviços Partilhados do Ministério da Saúde, E. P. P. E., estão relacionadas com a área da saúde, nomeadamente a aquisição viaturas médicas, suporte especializado e medicamentos.

Para finalizar a análise deste conjunto de 5000 anúncios, recorreu-se ao painel Previsão de Preços para obter valores dos preços de alguns dos anúncios que não apresentam preço-base.

Como seria de esperar (ver secção 2.5.1) a construção do modelo KNN foi de todas a mais rápida, sendo que o *Random Forest* em comparação com os outros modelos foi o que mais demorou. Como comprova a imagem 40, os erros de cada um dos modelos são muito semelhantes e elevados, indicando que apesar de todos os métodos usados no pré-processamento, as palavras que constituem as descrições dos anúncios não representam um bom conjunto de descritores para prever a variável-alvo, ou seja, não existe uma boa associação entre a "Descrição Sucinta"de um anúncio com o seu preço-base.

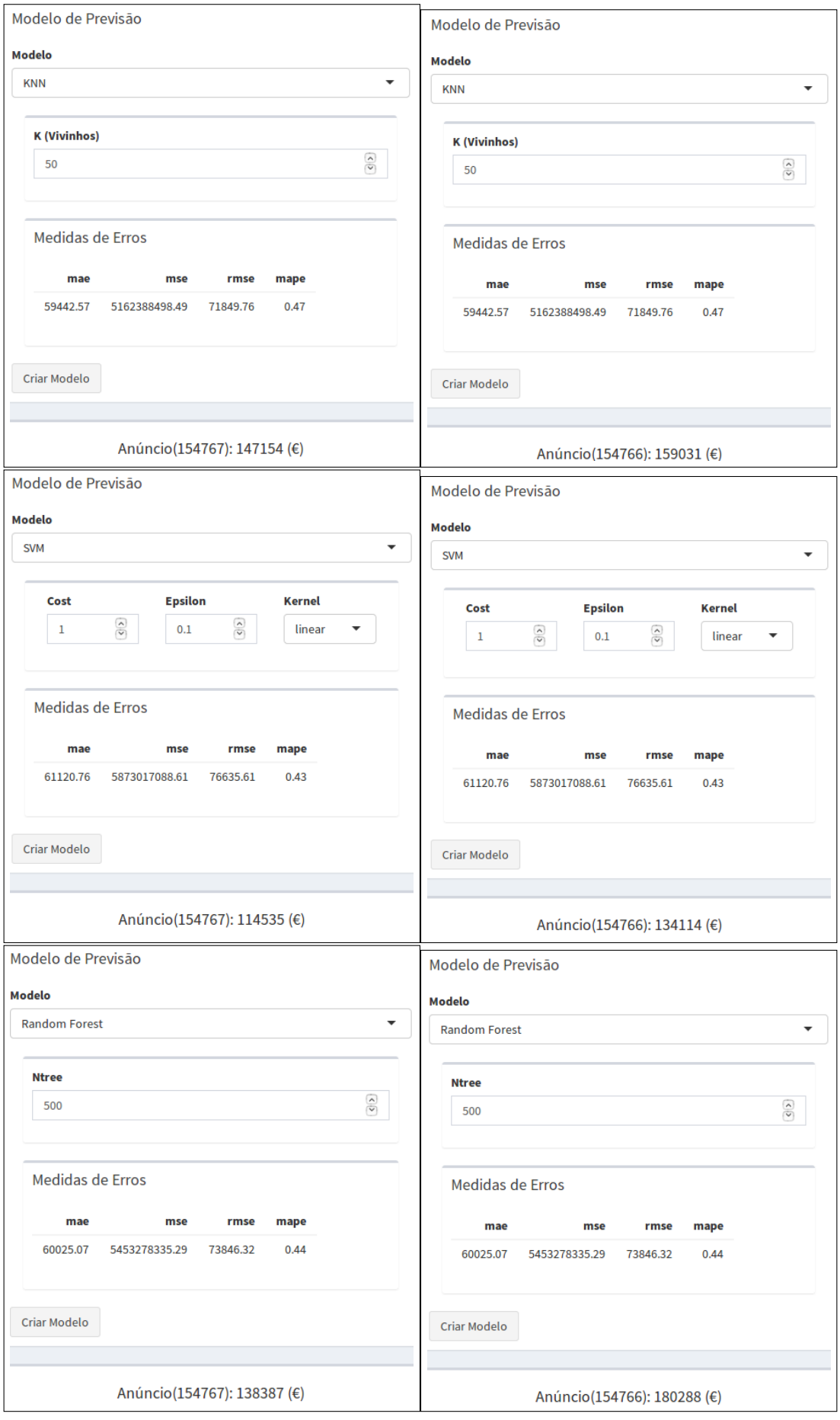


Figura 40: Previsões de preços de 2 dos 500 anúncios públicos portugueses que compõem o *corpus*

Capítulo 7

Conclusões e Trabalho Futuro

Analisados um conjunto de anúncios portugueses (ver secção 6) constatou-se que:

- Para um grande *corpus*, o algoritmo *YAKE* demora de forma significativa mais tempo do que o *RAKE*, uma vez que o primeiro envolve um conjunto de processos e cálculos mais complexos.
- Na análise geral que a *Cloudy* disponibiliza, além de a ferramenta garantir a fiabilidade dos dados, oferece ao utilizador um bom conhecimento sobre os mesmos com base nos diferentes campos dos anúncios;
- Através da análise geral pode-se ainda procurar pelo CPV pretendido, de forma a obter a sua distribuição de preços;
- As *wordclouds* conseguidas com os algoritmos *RAKE* e *YAKE* apresentam ao utilizador os temas dos anúncios tendo, como parâmetros os preços-base, mínimo e máximo.
- Uma vez que o *YAKE* extrai a mesma palavra em diferentes *keyphrases*, os resultados do algoritmo são, para este trabalho, melhores porque há um conjunto de termos importantes que este retorna ao contrário do *RAKE*.
- Apesar de mais demoradas, as *wordclouds* que devolvem as *keyphrases* dos anúncios revelam um maior conhecimento dos mesmos, sendo que por vezes é claro não só o tema de um determinado anúncio mas também o que se pretende adquirir ou realizar ao certo.
- Os modelos implementados apresentam erros bastante elevados e apesar, do KNN obter resultados melhores a diferença é quase impercetível.

No futuro, em primeiro lugar, seria indicado melhorar o desempenho da ferramenta, uma vez que a mesma tem um tempo de resposta bom quando o número de anúncios se encontra na casa das centenas, mas começa a reduzir significativamente na casa dos milhares. Como continuação deste trabalho, seria interessante desenvolver a *Cloudy*, para que esta permitisse uma leitura aumentada de contratos públicos portugueses. Uma vez que estes se encontram no BASE mas estão digitalizados, seria necessário o estudo da tecnologia OCR, em particular a biblioteca *tesseract* e a sua posterior implementação.

Referências

[1] Ah-Hwee Tan et al. Text mining: The state of the art and the challenges. In *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, volume 8, pages 65–70. sn, 1999. ii, iii

[2] James R Hollyer, B Peter Rosendorff, and James Raymond Vreeland. Democracy and transparency. *The Journal of Politics*, 73(4):1191–1205, 2011. 1

[3] https://www.een-portugal.pt/info/mercadounico/Paginas/default.aspx. 1

[4] http://www.base.gov.pt/mediaRep/inci/files/ccp2018/CCP-DL_111-B.pdf. 1

[5] https://ted.europa.eu/TED/main/HomePage.do. 1

[6] Jorge C. Leitão. Contratos e anúncios públicos em portugal. http://www.base.gov.pt, 2014. [Online; accessed 9-September-2018]. 1, 2, 18, 19, 22, 31

[7] Charu C Aggarwal and ChengXiang Zhai. *Mining text data*. Springer Science & Business Media, 2012. 3

[8] Text Mining Online. Getting started with keyword extraction. http://textminingonline.com/getting-started-with-keyword-extraction, 2015. [Online; accessed 9-January-2018]. 4

[9] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1–20, 2010. 4

[10] Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM, 1999. 6, 7

[11] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. A text feature based automatic keyword extraction method for single documents. In *European Conference on Information Retrieval*, pages 684–691. Springer, 2018. 7

[12] Viviane Orengo and Christian Huyck. A stemming algorithmm for the portuguese language. In *spire*, page 0186. IEEE, 2001. 9

[13] Olinda Nogueira Paes Cardoso. Recuperação de informação. *INFOCOMP*, 2(1):33–38, 2004. 9

[14] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):13, 2008. 9

[15] David Meyer and FH Technikum Wien. Support vector machines. *R News*, 1(3):23–26, 2001. 11

[16] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013. 11

[17] RStudio Team et al. Rstudio: integrated development for r. *RStudio, Inc., Boston, MA URL http://www. rstudio. com*, 42, 2015. 14

[18] Atlassian. Bitbucket. https://bitbucket.org. 14

[19] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2018. R package version 1.1.0. 14

[20] GOVIS. Código cpv. https://www.govis.pt/codigos/cpv. 20, 31

[21] The Apache Software Foundation. Rdata files. https://fileinfo.com/extension/rdata, 2010. [Online; accessed 9-August-2018]. 24

[22] Daniel Falbel. *ptstem: Stemming Algorithms for the Portuguese Language*, 2017. R package version 0.0.3. 27

[23] L. Torgo. An infra-structure for performance estimation and experimental comparison of predictive models in r. *CoRR*, abs/1412.0436, 2014. 27

[24] Fundação Francisco Manuel dos Santos. Direitos e deveres dos cidadãos. https://www.direitosedeveres.pt. [Online; accessed 9-September-2018].

[25] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[26] The Apache Software Foundation. Apache solr. http://lucene.apache.org/solr/features.html, 2010. [Online; accessed 9-January-2018].

[27] Daniel Lindsley. Pysolr. https://pypi.python.org/pypi/pysolr/3.5.0, 2009. [Online; accessed 13-January-2018].

[28] Scott Chamberlain. Solr: General purpose r interface to solr. https://cran.r-project.org/web/packages/solr/index.html, 2015. [Online; accessed 13-January-2018].

[29] Elasticsearch BV. Elastic search. https://www.elastic.co/products/elasticsearch, 2010. [Online; accessed 9-January-2018].

[30] Alex Ioannides. *elasticsearchr: A Lightweight Interface for Interacting with Elasticsearch from R*, 2018. R package version 0.2.2.

[31] Ryan Sonnek. Solr vs elasticsearch. http://blog.socialcast.com/realtime-search-solr-vs-elasticsearch, 2011. [Online; accessed 9-January-2018].

[32] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan, and Tyler Hunt. *caret: Classification and Regression Training*, 2017. R package version 6.0-78.

[33] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. R package version 1.6-8.

[34] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.