Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

Classificação de Documentos com Processamento de Linguagem Natural

Cedric Michael dos Santos

Mestrado em Informática e Sistemas

Coimbra, dezembro 2015

Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

Unidade Curricular Estágio/Projecto Industrial

Classificação de Documentos com Processamento de Linguagem Natural

Cedric Michael dos Santos

Orientador:

Prof. Doutor Carlos Manuel Jorge da Silva Pereira

Mestrado em Informática e Sistemas

Coimbra, dezembro 2015

Agradecimentos

A realização do projeto de mestrado contou com importantes apoios sem os quais este projeto não seria possível de realizar e aos quais estarei eternamente grato.

Gostaria de agradecer ao meu orientador do DEIS, Doutor Carlos Pereira, pelo acompanhamento, pela disponibilidade, pela colaboração no solucionar de dúvidas, pela partilha do saber, e pela possibilidade que me foi dada de realizar este projeto.

Agradeço aos meus colegas de curso, com os quais são partilhados conhecimentos, ideias, e pela força e motivação que partilhamos entre nós.

Finalmente, sou muito grato aos meus familiares pelo incentivo recebido não só durante a realização do projeto, mas em todos os anos em que estive no ensino superior.

Agradeço assim, a todos os intervenientes que participaram diretamente ou indiretamente à possibilidade de eu realizar o meu sonho em terminar o Mestrado em Informática e Sistemas.

Resumo

A quantidade de informação tem vindo a crescer ao longo dos anos, e a sua sobrecarga é hoje, um dos grandes problemas que as empresas e instituições enfrentam. A classificação dos documentos de forma automática surge então como uma necessidade, visto um sistema automático ser capaz de classificar milhares de documentos em apenas alguns segundos, o que seria impraticável de ser realizado por um ser humano. O maior desafio nesta área é obter os melhores resultados, maior eficiência computacional e maior capacidade de auto-aprendizagem, através nomeadamente da implementação de técnicas para a melhor seleção de *features*.

Este trabalho analisa e compara técnicas de classificação com processamento de linguagem natural, através da linguagem Python e com apoio da biblioteca para processamento de linguagem natural Natural Language Toolkit (NLTK). Neste projeto, foram implementados algoritmos para classificação de documentos, com técnicas de *Text Mining* e NLP, utilizando classificadores como Redes Neuronais, SVM's e Redes Bayesianas. Foram avaliados em vários casos de estudo e os resultados obtidos nos diferentes estudos atingiram valores médios acima de 90% de *accuracy.* Um dos principais objetivos deste projeto é aplicar os classificadores implementados na classificação de literatura na área das proteínas. Como resultado deste projeto, foi também implementada uma aplicação web para classificação de documentos, disponibilizando os algoritmos implementados.

Abstract

The amount of information has been growing over the years, and his overload is today one of the major problems that companies and institutions face. The automatic classification of documents is a necessity, since an automatic system would be able to classify thousands of documents in seconds, whereas a human could not accomplish the same task. The biggest challenge in this area is to get the best accuracy, greater computational efficiency and greater abality for self-learning, in particular through the implementation of techniques for the best selection of features.

This work analyzes and compares classification techniques with natural language processing through the Python language and library support from Natural Language Toolkit (NLTK). In this project were implemented document classification algorithms with Text Mining techniques and NLP, using classifiers such as neural networks, SVM's and Bayesian Networks. Several case studies were evaluated and the results obtained in the different studies reached average values above 90% accuracy. A major objective of this project is to apply the classifiers implemented in classification of proteins area literature. As a result of this project, it was also implemented a web application for documents classification providing the implemented algorithms.

Índice

Índice de Figuras

Índice de Tabelas

Abreviaturas

FN: False Negatives

FP: False Positives

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

IDE: Integrated Development Environment

JAVA EE: Java Enterprise Edition

JPA: Java Persistence API

JSP: JavaServer Pages

K-NN: K-Nearest Neighbors

KDT: Knowledge Discovery in Text

NLP: Natural Language Processing

NLTK: Natural Language Toolkit

ORM: Object-relational mapping

RBF: Radial Basis Function

SQL: Structured Query Language

SVM: Support Vector Machine

TF-IDF: Term Frequency-Inverse Document Frequency

TN: True Negatives

TP: True Positives

WEB: World Wide Web

WEKA: Waikato Environment for Knowledge Analysis

1 Introdução

1.1 Âmbito

O presente documento é realizado no âmbito da unidade curricular de Projecto Industrial, do 2º ano do Mestrado em Informática e Sistemas – Área de Especialização em Desenvolvimento de Software, ministrada no Departamento de Engenharia Informática e de Sistemas, do Instituto Superior de Engenharia de Coimbra.

O projeto teve início em Novembro de 2014 e terminou em Dezembro de 2015 (13 meses).

Com este projeto pretende-se construir uma aplicação web capaz de discriminar a classe de documentos, e que seja capaz de ser aplicado à classificação de literatura na área das proteínas. Na realização deste projeto pretendem-se usar técnicas de *Text Mining* e Processamento de Linguagem Natural (NLP – *Natural Language Processing*) de forma a tornar praticável o uso destas metodologias na classificação de documentos.

1.2 Objetivos do Projeto

Este projeto tem como principais objetivos específicos:

- Aprendizagem da linguagem de programação Python;
- Aprofundar competências no âmbito de implementação de aplicações para Internet;
- Aprendizagem sobre *Text Mining*, NLP e classificação de documentos com diversos classificadores inteligentes, tais como Redes Neuronais, SVM (*Support Vector Machine*) e Redes Bayesianas.

A necessidade de classificar documentos de forma automática tem vindo a crescer ao longo dos anos, sendo utilizada em muitas organizações de forma a evitar a utilização de recursos humanos na realização desta tarefa.

A plataforma a elaborar deverá tratar da classificação de documentos através de uma aplicação web com processamento de linguagem natural. O utilizador deve

poder inserir conjuntos de treino divididos em diferentes classes, de forma a que seja treinado um modelo e para que consecutivamente, o utilizador possa classificar novos documentos descaracterizados.

Os algoritmos de classificação de documentos devem ser implementados na linguagem de programação Python, utilizando as bibliotecas NLTK e Scikit-Learn, que disponibilizam métodos e técnicas de *Text Mining*, NLP e *machine learning*.

1.3 Motivação

Desde que surgiu na Idade do Bronze [1], a escrita tem vindo a evoluir sendo agora um meio indispensável de comunicação. Durante os últimos vinte anos, o número de documentos de texto em formato digital tem crescido imenso. O texto em linguagem natural faz parte do dia-a-dia atual e está presente em diversos meios, seja este escrito em papel, em páginas de internet, revistas, mensagens de e-mail, jornais entre outros.

A sobrecarga de informação é hoje um dos grandes problemas que as empresas e instituições enfrentam [2]. A triagem dos documentos úteis entre um conjunto alargado de documentos que não sejam do interesse desafia a criatividade e os recursos humanos das organizações. Da mesma forma, a triagem de documentos em categorias definidas pelas organizações, ao ser realizada manualmente faz com que seja uma tarefa árdua e longa, e não viável nomeadamente para grandes volumes de documentos.

Como consequência da necessidade de classificar documentos de forma automática, foram criados os sistemas de classificação de documentos. Estes sistemas existem em diversas aplicações com as seguintes funcionalidades principais: categorização de e-mails e remoção de spam, divulgação seletiva de informações aos consumidores de informação, classificação automática de artigos científicos, identificação do tipo de documento, atribuição da autoria de documentos, entre outras aplicações.

Existem diversas técnicas de processamento de linguagem natural para classificação de documentos, técnicas estas baseadas no facto dos documentos de diferentes categorias se distinguirem por *features* de linguagem natural contidas em

cada documento. As principais *features* para classificação de documentos podem ser descobertas a partir da estrutura das palavras, da frequência das palavras, e da estrutura da linguagem natural em cada documento.

1.4 Metodologia de Trabalho

Ao longo do projeto foram realizadas diversas reuniões de acompanhamento (geralmente reuniões semanais) com o orientador Doutor Carlos Pereira. Estas reuniões foram realizadas com o propósito de definir os métodos de trabalho a ser desenvolvido, acompanhar a evolução do projeto e na definição de objetivos semanais.

As tarefas realizadas durante a duração do projeto estão visíveis no seguinte diagrama de Gantt. Entre o mês de Junho e a segunda semana do mês Setembro, o aluno necessitou de estudar e realizar tarefas noutras unidades curriculares, pelo que foi realizada uma pausa na realização do projeto.

Figura 1 – Tarefas realizadas – Diagrama de Gantt

1.5 Estrutura do documento

Este relatório de projeto está dividido em 6 Capítulos, sendo eles: Introdução, Estado da Arte, Análise do Problema, Casos de Estudo, Aplicação Web e Conclusões.

Depois deste Capítulo introdutório de apresentação do projeto e dos seus objetivos, o segundo Capítulo, **Estado da Arte**, visa apresentar algumas

considerações sobre o tema, nomeadamente *Text Mining* e NLP, classificação de documentos e as suas fases. Neste Capítulo são também apresentados trabalhos relacionados utilizando *datasets* usados como casos de estudo neste projeto.

O Capítulo 3, **Análise do Problema**, tem como objetivo a análise do projeto, apresentando assim a visão, os requisitos, e a arquitetura de alto nível da aplicação.

No Capítulo 4, **Casos de Estudo**, são apresentados os exemplos de estudo realizados e a análise dos resultados obtidos nos mesmos. A implementação do algoritmo é explicada neste Capítulo.

O Capítulo 5, **Aplicação Web**, apresenta de forma detalhada a implementação da aplicação web, assim como da base de dados. São ainda apresentados os testes realizados.

O Capítulo 6, **Conclusões**, apresenta as conclusões obtidas na realização deste projeto. Este Capítulo encontra-se estruturado em três subcapítulos sendo eles: resultados do projeto, apreciação final e trabalho futuro.

2 Estado da Arte

Neste Capítulo pretende-se apresentar as técnicas e metodologias existentes no âmbito deste projeto, nomeadamente o *Text Mining*, NLP e as etapas de classificação de documentos. São ainda apresentados estudos de classificação de documentos com *datasets* usados neste projeto.

2.1 Text Mining

A informação disponível nos repositórios de dados é praticamente infinita, mas numa forma desordenada. A informação encontra-se disponibilizada e acessível geralmente através de texto, como por exemplo: revistas, jornais, artigos, páginas web, e-mails, e todo o tipo de documentação usada no dia-a-dia. Nos anos 90, surgiu a ideia de analisar estas "enormes quantidades de texto" para descobrir e gerar informação, encontrar padrões e anomalias em textos de forma automática. Assim na década de 90, a comunidade da Linguística Computacional considerou as grandes coleções de textos como um recurso a ser explorado de forma a produzir algoritmos para análise de textos [3].

Esta ideia era bastante atraente, nomeadamente para empresas de forma a organizar melhor os seus dados, apesar de ser bastante complicado de colocar o modelo em prática. Ao lidar com dados textuais é necessário lidar com informação não estruturada, ou seja, ao contrário dos dados estruturados em tabelas de base de dados, estes dados textuais não estão organizados em campos com tipos de valores. Assim sendo, comparando com dados armazenados em bases de dados relacionais, a informação em formato de texto é bem mais difícil de obter, tratar, e analisar para retirar informação. Podemos verificar a mesma informação de forma estruturada na tabela 1, e de forma não estruturada na figura 2.

Nome	Idade	Morada	Telefone
Cédric	25	Coimbra	910000000
José	29	Viseu	918888888

Tabela 1 – Informação estruturada

Olá, eu sou o Cédric, tenho 25 anos, moro em Coimbra e o meu número é o 910000000. O meu amigo José tem 29 anos, reside em Viseu e o número dele é o 918888888.

Figura 2 – Informação não estruturada

Surge assim *o Text Mining* (mineração de texto), uma subárea do *Data Mining* (mineração de dados), desenvolvida na procura de técnicas e processos para extração de informação e conhecimento a partir de dados não estruturados. Um dos grandes objetivos do *Text Mining*, técnica também conhecida como descoberta de conhecimento de textos (KDT – *Knowledge Discovery in Text*), consiste na análise de textos por parte do computador, e para tal, são usadas várias técnicas. Técnicas estas como a recuperação de informação (*information retrieval*), extração de informação onde são identificadas frases-chave e a sua relação com o texto, a sumarização que consiste na condensação do texto mantendo a informação, a classificação em que são atribuídas classes aos documentos, e o NLP que tem por objetivo construir sistemas que analisem, compreendem e gerem NLP [5]. O KDT é o processo de extrair conceitos explícitos e implícitos e as relações semânticas entre os conceitos, e tem como objetivo extrair conhecimento de grandes quantidades de dados de texto [4].

2.2 NLP

O Processamento de Linguagem Natural é uma subárea da inteligência artificial e consiste na aplicação de métodos e técnicas computacionais inteligentes que analisam e representam dados textuais e que possibilitam ao computador extrair a semântica da linguagem humana expressa em textos e voz. O NLP surgiu devido à necessidade da compreensão automática e comunicação em geral do ser humano com

o computador. O primeiro Sistema de NLP ocorreu em 1946, e consistiu na tradução automática de russo para inglês palavra a palavra [7].

Os sistemas de geração de linguagem natural convertem informação de bases de dados para linguagem natural, enquanto sistemas de compreensão de linguagem natural convertem a linguagem natural em dados estruturados, para permitirem a sua manipulação por parte dos computadores.

O NLP está direcionado essencialmente para os seguintes aspetos da comunicação em linguagem natural:

- **Som** - fonóloga;
- **Estrutura** - morfologia e sintaxe;
- **Significado** - semântica e pragmática.

O NLP trata-se de um mecanismo criado não só para extrair as informações de textos, mas também para facilitar a entrada de dados nos sistemas e a estruturação dos dados. O objetivo principal do NLP é processar linguagem humana para vários tipos de tarefas e aplicações [6], tais como [8]: **Extração de informação** – Procura informações diretamente nos textos, recolhendo a informação desejada. É baseado na busca de determinadas palavras-chave (*Tags*); Classificação de documentos – Técnicas que permitem classificar documentos a partir de uma determinada biblioteca de documentos; Tradução automática – Processo automático de tradução de um idioma para outro através do computador; Geração de resumos; Geração de linguagem natural; Interpretação de linguagem natural; Simplificação de texto; Correção ortográfica; Reconhecimento vocal.

Os sistemas de NLP são utilizados numa grande quantidade de domínios de estudo. O NLP é utilizado em diversas áreas, das quais se destacam o reconhecimento de entidades, segmentação de palavras e frases, remoção de *stopwords*, *stemming*, *part-of-speech tagging*, desambiguação e segmentação de texto.

O processo de NLP apresenta vários fases como se pode verificar na figura seguinte, o morfológico que lida com o tratamento das palavras, o léxico que se refere à análise do significado das palavras e de partes do diálogo, o sintático que trabalha a

gramática e a estrutura das frases, o fonético que lida com a pronúncia, o semântico que traduz o significado das palavras e frases, o discurso que lida com a estrutura de diferentes tipos de texto, e por fim, o pragmático que introduz conhecimento presente nas pessoas [9]. Dependendo do tipo de aplicação a ser construída, devem ser escolhidos quais destes níveis é indicado para utilização.

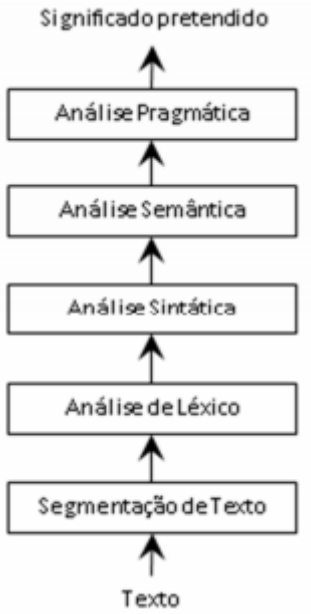


Figura 3 – Fases da análise de NLP

2.3 Trabalhos Relacionados

Neste subcapítulo pretende-se apresentar outros estudos realizados na classificação de documentos com *datasets* (Movie Reviews e Ohsumed) usados neste projeto. Os resultados destes estudos são apresentados de forma a comparar com os resultados obtidos neste projeto.

2.3.1 Movie Reviews

- **Classification of Movie Reviews using Character N-gram [45]**, apresenta um estudo utilizando *n-grams* de caracteres, com *n* variando entre 2 e 9. Neste trabalho foram usadas duas versões, uma utilizando a distância "*out-of-place*" e outra usando *Naive Bayes*. De seguida são apresentados os resultados com as *Naïve Bayes*, também utilizadas neste projeto:

gram-n	Correct % (accuracy)
2	57.75%
3	53.35%
4	57.15%
5	58.00%
6	57.55%
7	58.55%
8	59.15%
9	58.55%

Tabela 2 – Resultados do estudo Classification of Movie Reviews using Character N-gram

- **Fast and accuracte sentiment classification using an enhanced Naive Bayes model [46],** apresenta um estudo em Python, utilizando classificadores *Naive Bayes*: a versão original e a versão *Bernoulli Naive Bayes*. Neste estudo foi usado o "*Negation Handling*" em que, quando um termo é precedido por *not* ou *n't*, este é alterado para "not_+ termo". Esta técnica resolve o problema em que por exemplo a palavra "good" na frase "not good" iria contribuir para uma classificação positiva ao invés de negativa.

  Neste trabalho foram também usados as técnicas de *bigrams* e *trigrams*, e uma técnica de seleção de melhores *features*, *Mutual Information*.

  Os resultados obtidos neste estudo foram os seguintes:

Feature Added	Accuracy on test set
Original Naive Bayes	73.77%
Handling negations	82.80%
Bernoulli Naive Bayes	83.66%
Bigrams and trigrams	85.20%
Feature Selection	88.80%

Tabela 3 – Resultados do estudo Fast and accurate sentimento classification using an enhanced Naive Bayes model

- **Rotten Tomatoes: Sentiment Classification in Movie Reviews [47]**, apresenta um estudo com os classificadores *Naive Bayes*, SVM (linear) e SVM (rbf). Os testes foram usados utilizando *unigrams + bigrams*, *upweighting* que procura dar maior peso a *features* que estejam numa determinada posição do documento ou frase, e *tf-idf*. Os resultados (*accuracy*) são os seguintes:

Feature Added	Naive Bayes	SVM (linear)	SVM (rbf)
unigrams + bigrams	0.8395	0.7735	0.6665
sent. upweighting (last 2.0)	0.8440	0.7915	-
sent. & adj/adv upweight	0.8455	0.8090	-
tf-idf	0.8215	0.8460	0.8465
tf-idf + upweighting	0.8435	0.8690	0.8695
combined	-	0.8700	0.863

Tabela 4 – Resultados do estudo Rotten Tomatoes: Sentiment Classification in Movie Reviews

- **Sentiment Analysis and Classification of Online Reviews Using Categoical Proportional Difference [48]**, apresenta um estudo utilizando os classificadores SVM e *Naive Bayes* e a seleção de uma percentagem de melhores *features*, utizando CPD, IG e *Chi Square*. Os resultados (*accuracy*) com SVM foram os seguintes:

# of Features	Percentage	CPD	IG	Chi Square
36467	100	79.45	79.45	79.45
27350	75	79.05	79.45	77.5
17536	48	84.5	78.75	77.5
15204	41.7	88.1	78.75	77.5
9116	25	88.1	80.95	77.5
7037	19	88.1	82.2	65.2
3647	10	88.1	80.65	63.45

Tabela 5 – Resultados do estudo [48] com SVM

Os resultados (*accuracy*) com *Naive Bayes* foram os seguintes:

# of Features	Percentage	CPD	IG	Chi Square
36467	100	78.95	78.95	78.95
27350	75	79.05	79.45	77.2
17536	48	77.95	73.7	67.45
7190	19.7	77.95	67.1	67.45
3647	10	77.95	65.3	63.95

Tabela 6 – Rsultados do estudo [48] com Naive Bayes

- **Sentiment Classification of Movie Reviews by Supervised Machine Learning Approaches [49],** utiliza os classificadores SVM, *Baive Bayes* e k-NN com *cross-validation* (k=3). Os resultados (*accuracy*) obtidos foram os seguintes:

Numbe of reviews in training dataset	SVM	Naive Bayes	k-NN
50	60.07	56.03	64.02
100	61.53	55.01	53.97
150	67.00	56.00	58.00
200	70.50	61.27	57.77
400	77.50	65.63	62.12
550	77.73	67.82	62.36
650	79.93	64.86	65.46
800	81.71	68.80	65.44
900	81.61	71.33	67.44
1000	81.45	72.55	68.70

Tabela 7 – Resultados do estudo Sentiment Classification of Movie Reviews by Supervised Machine Learning Approaches

- **Sentiment Classification of Movie Reviews Using Contextual Valence Shifters [50]**, utiliza técnicas como o GI (*General Inquerer*) para identificar as palavras positivas e negativas, CTRW (dicionário de sinónimos), e *Enhanced*

que utiliza *bigrams* que contenham uma *feature* negativa ou "diminuidora". Os resultados foram os seguintes:

System	Accuracy
1) Basic: Term Couting, GI&CTRW&Adj	0.665
2) Basic: SVM, All Unigrams	0.849
Basic: Combined	0.854
1) Enhanced: Term Counting, GI&CTRW&Adj	0.678
2) Enhanced: SVM, All Unigrams + Bigrams	0.855
Enhanced: Combined	0.862

Tabela 8 – Resultados do estudo Sentiment Classification of Movie Reviews Using ContexturalValence Shifters

2.3.2 Ohsumed

- **Text Categorization with Support Vector Machines: Learning with Many Relevant Features [51]**, é um estudo que utiliza classificadores SVM, *Naive Bayes*, *Rocchio*, *Decision Tree* (C4.5) e k-NN. Os resultados apresentados são os melhores de entre os testes realizados com as 500, 1000, 2000, 5000 e 10000 melhores *features* e com todas as *features*. A métrica dos resultados é a *Precision*.

	Bayes	Rocchio	Decision Tree	k-NN	SVM (poly)	SVM (rbf)
Média Resultados	57.0	56.6	50.0	59.1	65.9	66.1

Tabela 9 – Resultados do estudo Text Categorization with Support Vector Machines: Learning with Many Revelelant Features

- **A Feature Weight Adjustment Algorithm for Document Categorization [52]**, apresenta uma solução para ajustar o peso das *features* mais discriminadoras das classes (FWA). Neste estudo usaram os seguintes classificadores: *Centroid-Based*, Rocchio (rocc), *Widrow-Hoff* (wh), SVM (rbf) e SVM (poly). De seguida são apresentados os resultados médios das 23 classes para cada classificador.

	rocc	wh	SVM (poly)	SVM (rbf)	centroid	FWA
Média Resultados	39.82	29.58	50.13	58.41	49.76	52.64

Tabela 10 – Resultados do estudo A Feature Weight Algorithm for Document Categorization

2.4 Classificação de documentos

A classificação de documentos tem como objetivo classificar documentos numa classe dentro de um conjunto de possíveis classes. Este processo pode ser realizado através de *Text Mining* e NLP, que permite descobrir informações desconhecidas, em textos de linguagem natural através de algoritmos computacionais.

O processo de classificação de documentos divide-se em diversas fases como se pode verificar na figura seguinte.

Figura 4 – Fases da classificação de documentos

A primeira fase consiste na obtenção dos dados, neste caso dos documentos, divididos nas suas respetivas categorias (classes).

A segunda fase permite realizar um pré-processamento do texto, através de vários operadores que são descritos nos próximos subcapítulos, como o *tokenizer*, remoção de *stopwords*, remoção de *non letters*, *transform cases*, *stemming*, *n-grams*, *prune method*.

A terceira fase é o processo de transformação do texto em *bag of words* (saco de palavras) ou num vetor modelo.

Na quarta fase são utilizados métodos de *Text Mining* como o *clustering*, classificação, *information retrieval*, entre outros.

A última fase permite avaliar e interpretar os resultados obtidos na fase anterior através de várias métricas, como a *precision*, *recall*, *accuracy*, *F-score / F-measure* entre outros.

2.4.1 Obtenção dos dados (documentos)

Os documentos para treino podem ser obtidos através de várias fontes, entre quais, bases de dados locais, *web services*, páginas web ou simplesmente através de diretorias na própria máquina. Nesta fase é importante os dados estarem organizados, de modo a que as classes ou categorias de classificação de cada documento seja identificada e assim obter os documentos divididos em categorias. Esta fase é necessária para a aprendizagem supervisionada.

A aprendizagem supervisionada é realizada a partir de exemplos, em que o sistema é auxiliado a construir o modelo, através das classes e dos exemplos (documentos) de cada classe. O sistema tem de descobrir semelhanças entre os documentos da mesma classe, para aprender, e de seguida conseguir descriminar a classe de novos documentos não usados na aprendizagem. Como exemplo para aprendizagem supervisionada, ao considerar duas categorias de documentos, positivos e negativos, os exemplos de documentos podem estar localizados em duas diretorias diferentes com os respetivos nomes das classes.

A aprendizagem não supervisionada ao contrário da aprendizagem supervisionada, é efetuada com base em observação e descoberta. Desta forma, as

classes não são conhecidas pelo sistema, pelo que o sistema deve observar os exemplos e reconhecer os padrões por si próprio de forma a descobrir as classes e ser capaz de classificar documentos de documentos não considerados na aprendizagem dos exemplos [10].

2.4.2 Pré-processamento de texto

A etapa de pré-processamento de texto, ou etapa de preparação do texto, é o primeiro passo nas técnicas de *Text Mining* e NLP para a estruturação do texto tendo um papel importante no processo e para os próximos passos. Esta fase visa remover dados desnecessários e organizar a informação extraída de forma estruturada de modo a que a classificação seja mais fácil e mais eficiente a processar. Algumas das técnicas de pré-processamento mais utilizadas são as seguintes: correção ortográfica, *tokenization*, transform case (*lower case)*, remoção de *stopwords*, *stemming*, *n-grams* e *prune percentual* (remoção de palavras) [11].

2.4.2.1 Correção Ortográfica

A correção ortográfica nos documentos permite eliminar os possíveis erros ortográficos do texto através de um corretor ortográfico utilizado juntamente com um dicionário de línguas, consoante a língua utilizada. Cada termo é então comparado com os termos do dicionário, e corrigido o que pode fazer com que palavras mal escritas que não entrariam nos termos importantes do documento a classificar passem então a fazer parte das palavras a serem tidas em conta na classificação. O principal objetivo desta técnica é evitar perder palavras-chaves na classificação.

2.4.2.2 Tokenization

Esta técnica é muito simples, e simplesmente permite que o texto seja dividido em *tokens* individuais. As frases dos documentos são assim "partidas" em *tokens* individuais removendo os espaços. Estes *tokens* servem então de entradas para os processos seguintes de *Text Mining*.

2.4.2.3 Transform Cases (Lower Case)

A técnica *lower case* é utilizada para transformar todas as palavras ou tokens em letras minúsculas de forma a que não sejam reconhecidas como tokens diferentes. Este método permite que o número de *features* seja reduzido.

2.4.2.4 Remoção de Non Letters

A remoção de *non netters* tem como objetivo eliminar todos os *tokens* que não sejam palavras. Nesta técnica pode também definir uma biblioteca de forma a remover outros termos irrelevantes consoante o *dataset*. Geralmente são removidos a pontuação, caracteres alfanuméricos e caracteres especiais.

2.4.2.5 Remoção de Stopwords

Nos documentos muitas palavras são usadas frequentemente, e que não ajudam a classificar documentos, e servem apenas para unir palavras nos textos. A frequência com que estas palavras são usadas, pode representar um obstáculo para a compreensão do conteúdo. Estas palavras são chamadas de não discriminantes ou *stopwords*. Como exemplo temos termos como “o”, “a”, “mas”, “ou”, “de”, ”que”, “em”, “no”, “para”, “por”, etc. Normalmente as palavras não discriminantes são proposições, substantivos e determinantes, conjunções, entre outras. Esta técnica permite remover estas palavras, fazendo com que a dimensão do número de palavras seja bastante diminuída.

2.4.2.6 Stemming

O *stemming* ou lematização é outra técnica utilizada para reduzir a lista de palavras indexadas num documento. Este método permite reduzir palavras que se encontram em formas derivadas para a sua forma base removendo as variações de palavras do tipo (plural, gerúndio, prefixos, sufixos, género e número) de modo que a palavra fique só com a raiz. Esta técnica tem duas grandes vantagens: o número de termos indexados é reduzido pois termos similares são mapeados como uma única entrada; a relevância dos resultados é muitas vezes melhorada significativamente.

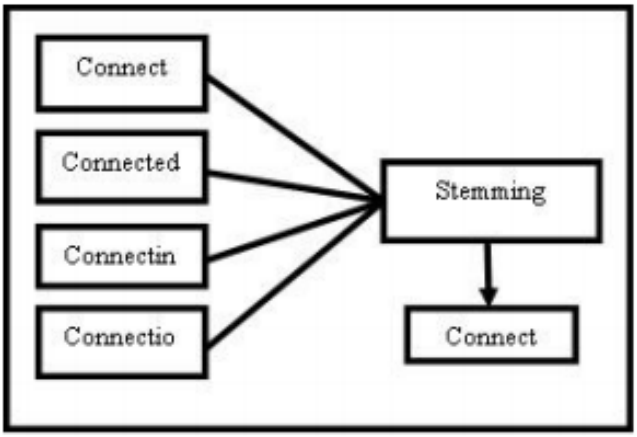


Figura 5 – Exemplo de Stemming

A figura acima é um exemplo da técnica de *stemming*, em que podemos ver quatro termos a serem todos reduzidos à sua forma base.

2.4.2.7 N-Grams

Os *n-grams* consistem na criação de uma sequência de palavras num comprimento *n*. Geralmente são usados *n-grams* de comprimento 2 também chamados de *bigrams*. Isto permite que os *tokens* indexados sejam termos de tamanho *n*. Esta técnica cria um número enorme de *tokens* (nº de tokens – 1) a somar aos *tokens* individuais, pelo que é conveniente depois de serem criados escolher apenas os *n-grams* de maior relevância, técnica esta explicada na aplicação do estudo no Capítulo 4.2.

Numa frase K com X palavras, o número de *n-grams* é igual a:

$$Ngrams_K = X - (N - 1)$$

Como exemplo deste processo, a frase "exemplo de classificação de documentos" são obtidos os seguintes *n-grams*: ["exemplo de", "de classificação", "classificação de", "de documentos"].

2.4.2.8 Prune Method

Esta técnica permite reduzir o número de termos indexados. O operador *prune method* especifica se palavras demasiadas frequentes ou demasiadas infrequentes devem ser ignoradas para a lista de palavras construída. Este método permite realizar a sua tarefa em frequências diferentes: percentagem, absoluta, ranking.

- **Percentagem**

Ignorar palavras que estejam presentes em menos ou mais de uma determinada percentagem escolhida, de entre todos os documentos da coleção.

- **Absoluta**

Ignorar palavras que estejam presentes em menos ou mais do que X documentos na coleção.

- **Ranking**

Especificar qual a percentagem das palavras mais frequentes ou menos frequentes que serão ignoradas.

2.4.3 Feature Extraction

Na fase de *feature extraction* os documentos são transformados na sua forma numérica, ou seja, cada documento é decomposto em termos e frequências representados em vetores. Cada termo é tratado como uma *feature*. Esta técnica é chamada de representação de *bag of words, bag of n-grams, ou* saco de palavras. Os documentos são descritos pelo número de ocorrências de palavras, ignorando a informação da posição relativa das palavras do documento [12]. O *bag of words* representa uma matriz, em que cada linha representa um documento e cada coluna representa um termo que ocorre no corpus, os valores de cada célula indicam a frequência de cada termo em cada documento.

Existem várias representações dos sacos de palavra, em que as principais são a representação binária, a representação por frequência de termo, e a representação por TF-IDF.

- **Representação Binária**

  A representação binária é a mais simples e onde a presença de um termo num determinado documento é representada de forma binária, isto é se um termo ocorrer uma ou mais vezes no documento, a célula correspondente da matriz é preenchida com o valor 1 ou True. Caso o termo não ocorra nenhuma vez no documento, a célula correspondente é preenchida com o valor 0 ou False. A fórmula é a seguinte:

$$x_{i,j} = \begin{cases} t_i \in d_j \Rightarrow 1 \\ t_i \notin d_j \Rightarrow 0 \end{cases}$$

  A tabela seguinte representa um exemplo de *bag of words* com representação binaria. Como se pode visualizar na tabela seguinte, na matriz apresentada a maioria dos valores dos atributos é zero, visto que na maioria das vezes um termo não aparece na maioria dos documentos. Pode-se verificar, por exemplo, que o termo "*accid*" ocorre pelo menos uma vez no documento 1 e no documento 5.

Documento	abondon	abil	absolut	absurd	academi	accent	accept	accid	accident	accompagn
1	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	1	0
5	0	0	0	0	0	0	0	1	0	0

Tabela 11 - Bag of Words com representação binária

- **Representação por Frequência de Termo**

  A representação por frequência de termo é relativamente idêntica à representação binária, onde a única alteração é que cada termo é representado pela frequência de ocorrências em cada documento. Nesta representação a célula correspondente é preenchida com o valor N, em que N é o número de ocorrências de um termo no documento. A fórmula é a seguinte:

$$w_{i,j} = ft_{i,j}$$

A tabela seguinte representa um exemplo de *bag of words* com representação por frequência de termo. Como se pode visualizar na tabela, na matriz apresentada a maioria dos valores dos atributos é zero, visto que na maioria das vezes um termo não aparece em muitos documentos. Neste exemplo, ao contrário do exemplo de representação binária, podemos visualizar quantas vezes um termo aparece num documento, no caso do termo "*absolut*", este ocorre duas vezes no documento 1828, uma vez no documento 1830 e nenhuma vez nos restantes documentos.

Documento	abondon	abil	absolut	absurd	academi	accent	accept	accid	accident	accompagn
1827	0	0	0	0	0	0	0	0	0	0
1828	0	0	2	2	0	0	0	0	0	0
1829	0	0	0	0	0	0	0	0	0	0
1830	0	0	1	0	0	0	0	0	0	0
1831	0	0	0	0	0	0	0	2	0	0

Tabela 12 - Bag of Words com representação por frequência de termo

- **Representação por TF-IDF**

  A representação por TF-IDF é uma das mais utilizadas, e permite verificar a importância do termo em relação aos outros. O processo é realizado a todos os termos para cada documento. Inicialmente, deve-se calcular a probabilidade de um termo aparecer no documento (TF), e de seguida é calculada a probabilidade inversa de um termo ocorrer no conjunto de documentos (IDF). Neste tipo de representação, um termo que ocorra com mais frequência num documento e poucas vezes no conjunto de documentos, é considerado um termo com maior peso ou importância. Um termo distribuído em toda a coleção de documentos, e que não esteja concentrado em pouco documentos, é considerado um termo com pouco ou nenhum poder de discriminação de relevância [13].

  A fórmula de cálculo é a seguinte:

$$(TF - IDF)_{i,j} = TF_{i,j} \times IDF_i$$

  TF é calculado com recurso à seguinte equação:

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Onde, $n_{i,j}$ representa o número de ocorrências do termo t_i no documento d_j . O denominador corresponde ao número total de termos do documento d_j.

IDF é calculado com recurso à seguinte equação:

$$IDF_i = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

Onde, $|D|$ é o número total de documentos do conjunto.

$|\{d : t_i \in d\}|$ é o número de documentos onde o termo t_i ocorre.

Considerando um conjunto de 2000 documentos, se um documento tiver 100 termos, e um termo ocorrer 5 vezes no documento e ocorrer em 90 documentos do *dataset*, o TF-IDF é calculado da seguinte forma:

$\text{TF} = 5 / 100 = 0.05$

$\text{IDF} = \log(2000/90) = 1.35$

$\text{TF-IDF} = 0.05 * 1.35 = 0.0675$

Na representação TF-IDF, podemos observar que [14]:

- Aumenta quando o termo ocorre várias vezes em poucos documentos
- Diminui quando a frequência do termo é baixa
- Diminui quando o termo ocorre em muitos documentos
- Torna-se nulo quando ocorre em todos os documentos

A tabela seguinte representa um exemplo de *bag of words* com representação por TF-IDF.

Documento	abondon	abil	absolut	absurd	academi	accent	accept	accid	accident	accompagn
1827	0	0	0	0	0	0	0	0	0	0
1828	0	0	0.128	0.186	0	0	0	0	0	0
1829	0	0	0	0	0	0	0	0	0	0
1830	0	0	0.068	0	0	0	0	0	0	0
1831	0	0	0	0	0	0	0	0.167	0	0

Tabela 13 - Bag of Words com representação por TF-IDF

2.4.3.1 Seleção das Melhores Features

Esta técnica é uma das mais importantes, e que aumenta mais a qualidade dos resultados obtidos no treino de classificadores. Quando o modelo de classificação tem uma grande quantidade de *features*, como é o caso da maioria dos problemas de classificação documentos, mesmo após o processo de pré-processamento de texto, a maioria das *features* são de baixa informação, ou seja, não são suficientemente categorizadoras de uma determinada classe. Estas *features* são comuns a todas as classes e portanto contribuam com pouca informação para o processo de classificação. Uma grande quantidade de *features* de baixa informação, faz com que os resultados obtidos diminuam de qualidade e de desempenho. As *features* de baixa informação, ao serem eliminadas do modelo, permitem que sejam usadas apenas as *features* de alta informação, reduzindo a quantidade de *feature*, o que permite aumentar o desempenho e a utilização menor da memória para o treino do classificador [15].

Para encontrar as *features* mais discriminadoras ou mais classificadoras das suas classes, é necessário calcular o ganho de informação de cada palavra. O ganho de informação é uma medida que calcula o quão comum uma *feature* está representada numa determinada classe, em comparação com o quão comum está representada nas restantes classes. No *dataset* NLTK Movie Reviews, que contém as classes "positivo" e "negativo", uma palavra que ocorra principalmente em comentários positivos e raramente em comentários negativos, é uma palavra com alta informação. Por exemplo, a palavra "excelente", ao que tudo indica, é uma palavra de alta informação para a classe "positivo".

Umas das melhores métricas para o cálculo do peso de cada *feature* são o *Chi Square* e o *Information Gain*.

- **Chi Square**

  Esta métrica mede a falta de independência entre o termo *i* e a classe *c* e pode ser comparada à distribuição X^2 com um grau de liberdade para julgar extremos. Usando uma tabela de contingências de um termo *i* e uma categoria *c*, onde *w* é o número de vezes que *i* e *c* coocorrem, *y* é o número de vezes que *i* ocorre sem *c*, *x* é o número de vezes que *c* ocorre sem *i*, *z* é o número de vezes que nem *i* nem *c* ocorrem [16]. A fórmula para cálculo é a seguinte:

$$fw_{ic} = \frac{(wz - xy)^2}{(w + x)(w + y)(x + z)(y + z)}$$

- **Information Gain**

  Esta métrica permite descobrir o peso que um determinado termo tem, na discriminação entre classes previamente conhecidas. Esta técnica verifica o grau de correlação existente entre cada classe [16]. A fórmula é a seguinte:

$$fw_{ic} = H(C) - H(C|W_i) = \sum_{C_s \in \{C, \overline{C}\}} \sum_{W_t \in \{W_i, \overline{W}_i\}} p(c_s, w_s) \log \frac{p(c_s, w_t)}{p(c_s)p(w_t)}$$

  Onde, p(c) é o número de documentos que pertencem à classe c dividido pelo número de documentos, e p($\overline{w}$) é o número de documentos sem o termo w dividido pelo número total de documentos.

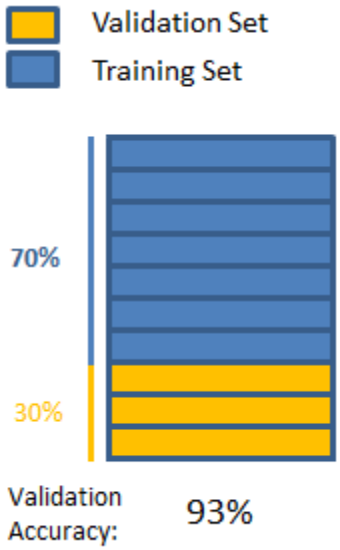
2.4.4 Classificação

A classificação dos documentos é realizada através de classificadores, que consistem em treinar os mesmos com um conjunto de treino. De seguida é aplicado ao classificador um conjunto de testes de forma a obter os resultados da classificação e respetiva avaliação obtida através da matriz de confusão devolvida pelo classificador, matriz esta que possui os dados sobre os Verdadeiros Positivos, Falsos Positivos, Verdadeiros Negativos e Verdadeiros Negativos (Capítulo 2.4.5).

Os documentos usados como conjuntos de treino e de teste podem ser obtidos de diversas formas, nomeadamente por percentagem treino/teste ou por *cross-validation*:

- **Percentagem treino/teste**

Neste caso, o conjunto de treino representa geralmente 70% dos documentos enquanto o conjunto de teste é representado pelos restantes documentos, 30%. A decomposição em 80% para treino e 20% para teste também é frequentemente utilizada [17][18]. Neste projeto foi utilizado o rácio 70% / 30%. De forma a que o modelo de classificação obtenha melhores resultados, o conjunto de treino é sempre maior que o conjunto de teste, embora utilizando mais documentos de teste faz com que a estimativa de erro seja mais precisa. Os conjuntos de treino e de teste podem ser escolhidos aleatoriamente, ou definindo os primeiros 70% de documentos para treino e os últimos 30% para teste. Este método de percentagem é geralmente mais utilizado em *datasets* com bastantes documentos. A figura seguinte representa os conjuntos de treino e teste em que devem ser treinados 70% documentos de cada classe e testados 30% de cada classe.



Fonte: https://chrisjmccormick.wordpress.com/2013/07/31/k-fold-cross-validation-with-matlab-code/

Figura 6 – Percentagem treino 70% / teste 30%

- **Cross-Validation**

O *cross-validation* ou validação cruzada consiste na divisão dos dados em vários subconjuntos de teste e treino. Existem várias técnicas *de cross-validation*, entre quais o método ***leave-one-out*** e o método ***k-fold***.

O método ***leave-one-out*** é considerado um método exaustivo em que gera um grande esforço computacional e é indicado para situações onde poucos dados para treino estão disponíveis. Este método define o número de subconjuntos *K* igual ao número de exemplos presentes no conjunto de treino e teste. Neste caso são realizados *K* classificações e em cada uma, o conjunto de teste representa apenas um documento, sendo o conjunto de treinos representado pelos restantes documentos [19].

O método ***k-fold*** permite dividir o conjunto de documentos em *K* subconjuntos mutuamente exclusivos do mesmo tamanho. Destes *K* subconjuntos de documentos, para cada iteração um é retido para ser utilizado como conjunto de testes, e os restantes *K-1* conjuntos são retidos para servirem de conjuntos de treino. Este processo de validação cruzada é repetido *K* vezes de modo a que cada um dos *K* subconjuntos seja utilizado uma vez como dado de teste para a avaliação do modelo. O resultado final do processo é representado pela média do desempenho do classificador nas *K* iterações. Quando o *K=Número de documentos* o método é considerado como o *leave-one-out*, em que este pode ser definido como uma variação do *k-fold* [19][20].

O método k-fold possui ainda outra variante chamada de ***stratified k-folds*.** O *stratified k-folds* garante que os *k-folds* são construídos preservando a percentagem de amostras para cada classe. Numa classificação de duas classes, este método garante que cada *fold* contenha aproximadamente as mesmas proporções de ambas as classes [20][21].

O principal objetivo de se utilizar o *cross-validation* e os testes múltiplas vezes é aumentar a confiabilidade da avaliação da precisão do classificador.

A imagem seguinte ilustra o funcionamento do método de *cross-validation* com *k-fold* com *K=10*.

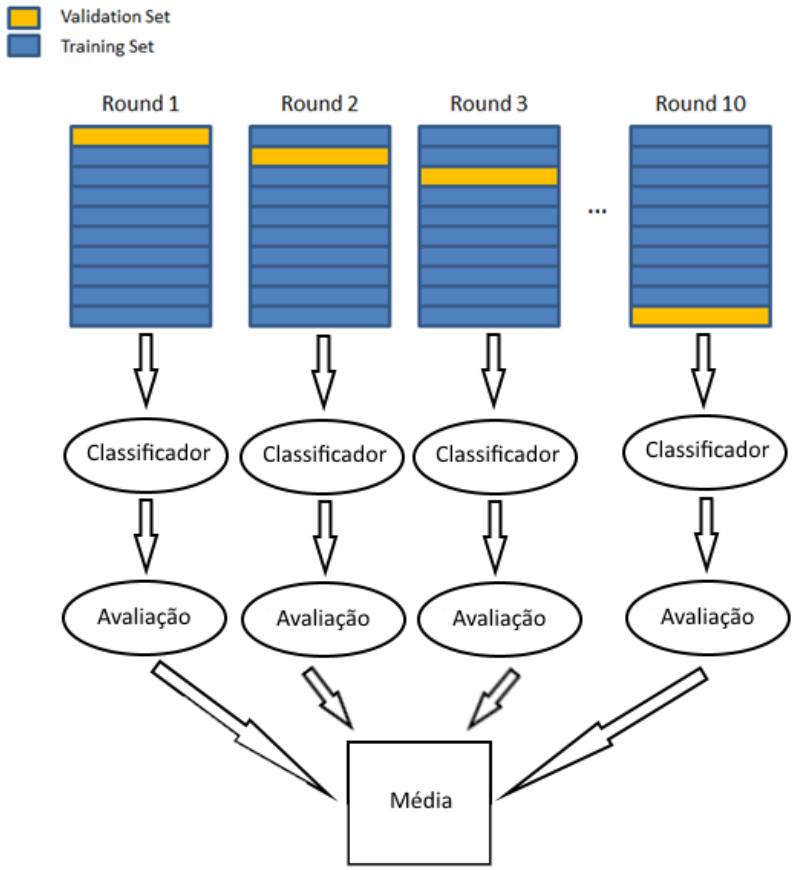


Figura 7 – Cross-Validation k-fold = 10

Nos casos de estudo (Capítulo 4), foram utilizados diferentes classificadores de forma a testar qual resulta em melhores resultados. Os classificadores utilizados foram os seguintes: Decision Tree, k-NN (*K-nearest neighbors*), *Logistic Regression* ou *Maximum Entropy*, *Naive Bayes*, RBF (*Radial Basis Function*), Redes Neuronais, e SVM.

De seguida são apresentados os classificadores usados nos casos de estudo.

2.4.4.1 Árvore de decisão

O algoritmo árvore de decisão (*decision Tree*) reconstrói a categorização manual dos documentos de treino através da construção de *queries* verdadeiras/falsas sob a forma de uma estrutura de árvore. Na estrutura da árvore de decisão, as folhas representam a categoria correspondente de documentos e os ramos representam conjunções de *features* que levam a essas categorias ou classes. A classificação do

documento é realizada colocando o documento no nó da raiz da árvore e deixando o mesmo correr através da estrutura de *queries* até atingir uma certa folha, o que representa a categoria para a classificação do documento.

A tarefa de classificação de documentos, geralmente envolve um grande número de *features* relevantes, e portanto quando aplicado em grandes quantidades de documentos, este classificador por levar a mau desempenho. Este método classifica de modo eficaz os dados de treino, no entanto a classificação de novos documentos apresenta um mau desempenho. No entanto, quando o número de atributos é menor, o desempenho, simplicidade e compreensibilidade das árvores de decisão são as principais vantagens deste método [22].

2.4.4.2 k-Vizinhos mais próximos (k-NN)

O algoritmo k-NN (*k-Nearest Neighbors*) é um método baseado em distâncias. Este algoritmo varia consoante o número de vizinhos considerados, ou seja, varia consoante o valor de *k*.

Cada objeto representa um ponto num espaço que é definido pelos atributos, chamado espaço de entrada. Definindo uma métrica nesse espaço é possível calcular a distância entre dois objetos.

Com este método, um documento é classificado na mesma categoria que a maioria dos k vizinhos mais próximos.

A fase de treino é baseada em exemplos de documentos de treino de cada classe. Assim, tal como na categorização linear, quanto mais exemplos de treino forem treinados, melhor será o resultado [26].

O algoritmo passa pelas seguintes etapas:

1. Calcular quais os *k* documentos mais próximos (*k* vizinhos mais próximos segundo a distância escolhida), e as categorias a que pertencem.
2. Atribuir o documento à categoria que tem mais representantes entre os *k* vizinhos mais próximos (cada vizinho vota numa classe).

2.4.4.3 Naive Bayes

O classificador *naive bayes* é um método probabilístico com base no Teorema de Bayes. Este classificador assume que os valores dos atributos de um exemplo são independentes da sua classe, ou seja, a probabilidade de um evento A (que pode ser uma classe), dado um evento B (que pode ser o valor do conjunto de atributos de entrada), não depende apenas da relação entre A e B, mas também da probabilidade de observar A independentemente de observar B.

A predição da probabilidade de um conjunto de *features* pertencer a uma determinada classe (*label*), é dada segundo a fórmula seguinte [24]:

$$P(label|features) = P(label) \times P(features|label) / P(features)$$

As descrições dos vários parâmetros da fórmula anterior são descritas de seguida:

- **P(label)**

  Probabilidade à priori da ocorrência de uma classe, onde a probabilidade é de que um conjunto de *features* aleatório irá pertencer a essa classe. Isto é baseado no número de instâncias de treino com essa classe comparado com o número total de instâncias de treino. Por exemplo, se 60 de entre 100 instâncias de treino pertencer à classe, a probabilidade à priori da classe é de 60%.

- **P(features | label)**

  Probabilidade à priori de um determinado conjunto de *features* ser classificado nesta classe. Isto baseia-se nas *features* que ocorreram em cada classe nos dados de treino.

- **P(features)**

  Probabilidade à priori de um determinado conjunto de *features* ocorrer. É a probabilidade de um conjunto aleatório de *features* ser o mesmo que o conjunto do conjunto se *features* dado. É baseado no conjunto de *features* observado nos dados de treino. Por exemplo, se o conjunto de teste dado ocorrer duas vezes em 100 casos de teste, a probabilidade à priori é de 2%.

- **P(label | features)**

Probabilidade de que as *features* dadas devem ter essa classe. Se este valor for alto, então podemos estar razoavelmente confiantes de que a classe está correta para as *features* dadas.

Existem várias variantes do *Naive Bayes*, o NLTK disponibiliza o classificador NaiveBayesClassifier, enquanto o Scikit-learn disponibiliza o MultinomialNB e o BernoulliNB. A principal diferença entre o NaiveBayesClassifier e o MultinomialNB, é que o MultinomialNB pode trabalhar com valores de *features* distintos, como a frequência das palavras, enquanto o NaiveBayesClassifier do NLTK assume um pequeno conjunto de valores das *features*, tais que strings ou booleanos. O BernoulliNB, pode também trabalhar com diferentes valores, tornando estes valores como binários, em que os valores finais são 0 ou 1.

2.4.4.4 Regressão Logística

O algoritmo regressão logística (*logistic regression*), disponibilizado pelo Scikit-Learn também pode ser denominado *maximum entropy* (NLTK). O classificador *logistic regression* utiliza um modelo muito semelhante ao modelo utilizado pelos classificadores *naive bayes*. Ao contrário dos *naive bayes* que utilizam probabilidades para os parâmetros do modelo, o *kogistic regression* utiliza técnicas de busca para encontrar um conjunto de parâmetros que maximizam o desempenho do classificador. O classificador olha para o conjunto de parâmetros que maximiza a probabilidade total do corpus de treino, definido pela seguinte fórmula:

$$P(features) = \sum_{x \in corpus} P(label(x)|features(x))$$

Onde $P(label|features)$, a probabilidade de uma entrada cujas *features* são *features* que terão a classe *label*, que é definida como:

$$P(label|features) = P(label, features) / \sum_{label} P(label, features)$$

Não há nenhuma maneira de calcular diretamente os parâmetros do modelo que maximizam a probabilidade do conjunto de treino. Portanto, o classificador escolhe os parâmetros do modelo utilizando técnicas de otimização iterativa, que

inicializam parâmetros do modelo com valores aleatórios, e de seguida, redefine repetidamente esses parâmetros de forma a se aproximarem o mais perto possível da solução ótima. As técnicas de otimização garantem que os parâmetros se vão aproximar dos valores ideais, mas não vai necessariamente fornecer um meio de determinar quando esses valores ótimos foram alcançados. Devido aos parâmetros serem selecionados por técnicas de otimização iterativas, o classificador pode levar muito tempo para aprender, nomeadamente quando o tamanho do conjunto de treino, o número de *features* e o número de classes são grandes.

O NLTK disponibiliza dois modelos do classificador *maximum entropy*: o *Generalized Iterative Scaling* (GIS) e o *Improved Iterative Scaling* (IIS). Estes algoritmos são mais lentos e têm piores resultados do que o *logistic regression* do Scikit-learn. Isto pode ser explicado pelo Scikit-learn se focar na otimização do processamento numérico utilizando a biblioteca NumPy [24].

2.4.4.5 Redes Neuronais

O algoritmo redes neuronais é um método de classificação baseado em otimização inspirado pela estrutura biológica do cérebro humano, o responsável pelo processamento de diversas informações e geração de respostas. O algoritmo é desenvolvido tendo em conta a estrutura e funcionamento do sistema nervoso, sendo o objetivo simular a capacidade de aprendizagem do cérebro humano na aquisição de conhecimentos [26].

As Redes Neuronais são compostas por unidades de processamento simples interconectadas. Estas unidades são conhecidas como neurónios artificiais e estão dispostas por uma ou mais camadas e interligadas por um elevado número de conexões. Essas conexões possuem pesos que ponderam a entrada recebida por cada neurónio. Os pesos são obtidos através de um procedimento de aprendizagem realizado por um processo de *back-propagation*, que implica a comparação da saída da rede neuronal com a saída para a qual a rede foi concebida. Esta diferença faz com que os pesos das conexões entre os neurónios da rede sejam alterados, o que permite à rede aprender e ir ficando mais precisa. Uma vez que a rede está suficientemente calibrada, esta atinge um ponto em que não é mais necessário treiná-la.

A arquitetura de uma rede neuronal é definida pelo número de unidades de processamento e à forma como os neurónios estão conectados.

As redes neuronais na classificação de documentos, pode ter um alto custo computacional a nível de CPU e de memória [22], e podem demorar mais tempo a treinar do que os restantes classificadores.

As redes neuronais podem ser aplicadas em vários problemas reais [25]:

- **Classificação**

  Uma das potencialidades das redes neuronais e das mais usadas consiste na sua utilização para classificar padrões de entradas em diferentes categorias a partir de imagens, texto, eletrocardiogramas entre outros.

- **Redução de ruído**

  A rede neuronal pode ser treinada de modo a reconhecer um determinado número de padrões de entradas, reproduzindo-os nas suas saídas. Desta forma pode ser utilizada para reproduzir o padrão original quando este é apresentado com ruído.

- **Predição**

  A predição consiste em prever o valor futuro de uma variável a partir de valores históricos dessa variável ou de outras variáveis relacionadas com ela.

A imagem seguinte mostra um exemplo da composição de uma rede neuronal.

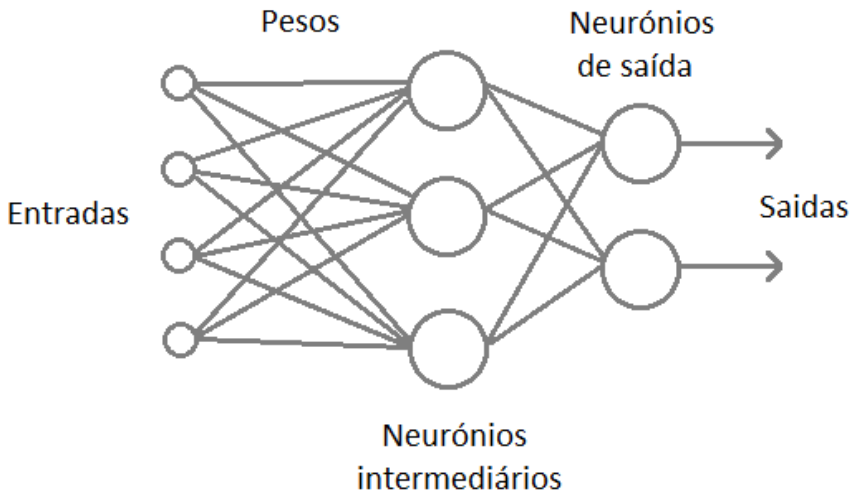


Figura 8 - Rede Neuronal

2.4.4.6 Rede Neuronal Função de base radial (RBF)

No RBF (*Radial Basis Function*), a aprendizagem é realizada encontrando uma superfície num espaço multidimensional, que fornece o melhor ajuste para os dados de treino. A generalização é encarada como a aplicação dessa superfície para interpolação do conjunto de dados de teste.

O RBF é constituído por três camadas [27]:

- **Camada de entrada**

A camada de unidades de entrada onde há um neurónio para cada entrada.

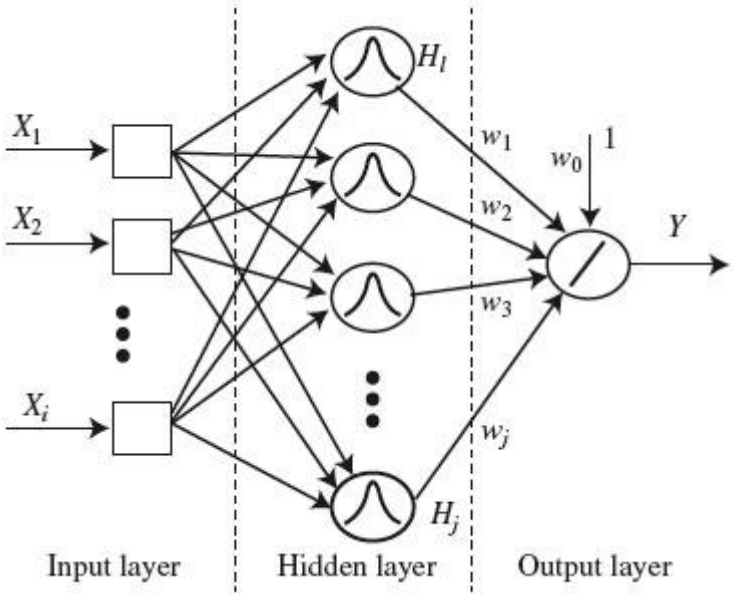
- **Camada interna**

A camada interna (ou camada oculta) não-linear, onde o número de neurónios é variável e a quantidade ótima é obtida no treino. Cada neurónio consiste numa função de base radial que fornece um conjunto de funções que constituem uma base arbitrária para os padrões de entrada e representa uma dimensão.

- **Camada de saída**

A camada de saída linear.

A imagem seguinte representa as camadas do RBF.



Fonte: André Bianconi, Cláudio J. Von Zuben, Adriane B. de S. Serapião, José S. Govone.(2010). Artificial neural networks: A novel approach to analysing the nutritional ecology of a blowfly species, Chrysomya megacephala

Figura 9 – Rede Neuronal RBF

2.4.4.7 Máquina de Suporte Vetorial (SVM)

Os SVM (*Support Vector Machines*) são classificadores disponibilizados pelo Scikit-learn, através de três versões diferentes: SVC (*Support Vector Classification*), NuSVC (*Nu-Support Vector Classification*) e LinearSVC (*Linear Support Vector Classification*).

O SVM é um método baseado em otimização, tal como as redes neuronais. O algoritmo resolve problemas de classificação lineares e não-lineares. Na classificação de conjuntos de dados linearmente separáveis ou que possuam uma distribuição aproximadamente linear são utilizados SVM's lineares. Quando os dados não são lineares é necessário um SVM não linear.

Os SVM's não lineares mapeiam o conjunto de treino do seu espaço original para um novo espaço de maior dimensão, designado de espaço de características através do teorema de Cover. O teorema afirma que dado um conjunto de dados não lineares num espaço de entradas X, então X pode ser transformado num espaço de características em que os objetos são linearmente separáveis com alta probabilidade. A informação necessária para o mapeamento é obtida através de funções designadas de *kernels*. As funções *kernels* retornam o cálculo de produtos escalares entre os

objetos no espaço de características. As funções *kernel* recebem dois pontos no espaço de entradas e calcula o produto escalar desses objetos no espaço de características. Existem vários tipos de *kernels*, e entre os mais utilizados estão os polinomiais, os de base radial (RBF) e os sigmoidais [28].

Após o mapeamento, é aplicado o SVM linear sobre o novo espaço, em que o classificador encontra o hiperplano com maior margem de separação. O mapeamento é exemplificado na figura seguinte [22].

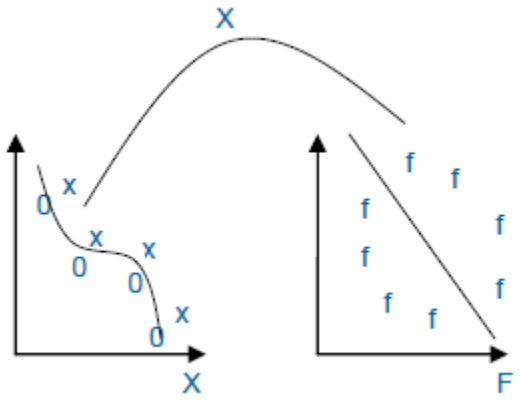


Figura 10 – Mapeamento não linear para espaço de características

O algoritmo SVC do Scikit-learn é implementado com base na biblioteca *libsvm* e é um SVM não linear onde pode ser definido o tipo de *kernel* como parâmetro [29]. O algoritmo LinearSVC é um algoritmo de SVM linear, implementado com base na biblioteca *liblinear*, o que permite mais flexibilidade na escolha das penalidades e tem melhores resultados quando o número de amostras é elevado [30]. O NuSVC é semelhante ao SVC, mas utiliza um parâmetro para controlar o número de vetores de suporte [31].

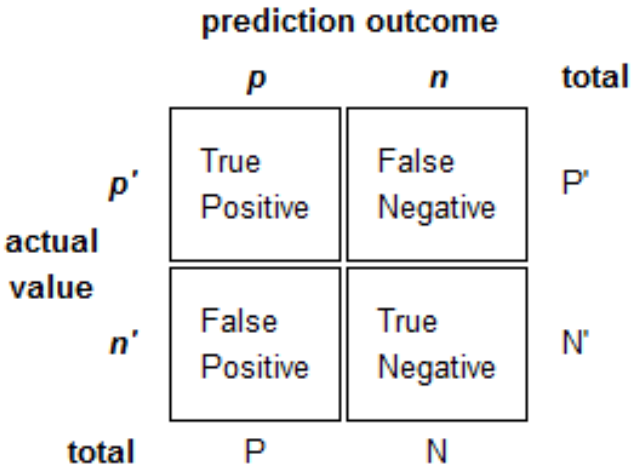
2.4.5 Avaliação dos resultados

Após a classificação dos documentos através dos classificadores, é necessário validar a qualidade da solução através da avaliação dos resultados, de forma a verificar se o classificador realizou as previsões corretamente. Com base no conjunto de treino e no conjunto de teste, é comparado o valor da classe de cada exemplo no

conjunto de teste com os resultados que se obtém na previsão. Os resultados são obtidos através de uma matriz de confusão.

2.4.5.1 Matriz de confusão

A matriz de confusão apresenta os resultados da classificação. As linhas representam as instâncias previstas de uma classe enquanto as colunas representam as instâncias reais de uma classe. Estas matrizes são muito fáceis de analisar quando o sistema é composto por duas classes. No caso de haver mais classes, estas podem ser reduzidas a duas, a classe alvo (classe positiva), e as restantes podem ser agrupadas para formar uma só classe (classe negativa) tal como foi realizado no caso de estudo que se encontra em anexo (Anexo G - Resultados de classificação de documentos com Python – 2ª versão (Ohsumed)).



Fonte: http://tex.stackexchange.com/questions/20267/how-to-construct-a-confusion-matrix-in-latex

Figura 11- Matriz de confusão

Resumindo, e como se pode verificar na figura anterior, podemos considerar a matriz de confusão como uma tabela com duas linhas e 2 colunas que regista o número de *True Negatives* (TN), *False Positives* (FP), *False Negatives* (FN) e *True Positives* (TP) [32].

- **TN** – Instâncias negativas classificadas como negativas
- **FP** – Instâncias negativas classificadas como positivas

- **FN** – Instâncias positivas classificadas como negativas
- **TP** – Instâncias positivas classificadas como positivas

Na tabela seguinte, está representada uma matriz de confusão, onde temos:

TP = 722; TN = 768; FP = 232; FN = 278

	true pos	true neg
pred. pos	722	232
pred. neg	278	768

Tabela 14 – Exemplo de matriz de confusão

2.4.5.2 Accuracy, Recall, Precision, F-measure

O *accuracy*, *recall*, *precision* e *F1-measure* são medidas para a avaliação dos resultados e são apresentadas de seguida [33].

- **Accuracy**

  Accuracy é a fração de documentos corretamente classificados entre o total dos registos. A fórmula para o cálculo desta medida é a seguinte:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

- **Recall**

  O *recall* mede a capacidade do sistema em encontrar o que realmente se pretende e pode ser considerada uma medida de completude. Esta medida é obtida através da fórmula seguinte:

$$Recall = \frac{TP}{FN + TP}$$

- **Precision**

Precision é uma medida que mede a proporção de exemplos positivos classificados corretamente entre todos aqueles preditos como positivos. Esta medida mede a capacidade do sistema em rejeitar o que não se pretende e é considerada uma medida de precisão ou de fidelidade. A fórmula de cálculo é a seguinte:

$$Precision = \frac{TP}{FP + TP}$$

- **F1-Measure**

F1-Measure é uma medida de teste de precisão e é definida como a média harmónica entre os valores de *recall* e do *precision* através da seguinte fórmula:

$$F1 - Measure = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

3 Análise do Problema

3.1 Visão e Âmbito

O sistema de classificação tem como principal objetivo a classificação de documentos em categorias, e deve permitir classificar um conjunto de documentos dentro de duas ou mais classes definidas pelo utilizador. Para realizar as tarefas de classificações, o utilizador deve usar uma aplicação web, que de alguma forma pode representar uma comunidade de utilizadores em que podem ser utilizados *datasets* inseridos por outros utilizadores caso o utilizador que o tenha inserido o permita.

Dada a quantidade considerável de dados da aplicação, nomeadamente a nível de utilizadores, classificadores, informações dos *datasets*, entre outros dados, o sistema necessita de armazenar uma quantidade de informação elevada. Assim, é necessário criar uma base de dados, para guardar os dados. Os *datasets* carregados pelos utilizadores são armazenados no próprio servidor.

O processo de categorização dos documentos é uma tarefa realizada por Scripts em Python utilizando as bibliotecas adequadas para processamento de linguagem natural e *machine learning*, caso do NLTK e Scikit-Learn (Capítulo 4.1.1).

O sistema deve assim, interligar a aplicação web que serve de interface para o utilizador, com os scripts necessários à classificação dos documentos.

3.2 Requisitos Funcionais

Após a definição do problema e das características do sistema, são definidas as funcionalidades da aplicação web, tendo esta como objetivo principal a categorização de documentos. O diagrama de casos de uso e a escrita dos mesmos estão disponíveis em anexo (Anexo A – Casos de Uso). Os principais diagramas de atividade estão igualmente disponíveis em anexo (Anexo B – Diagramas de Atividade).

A aplicação web permite ser usada por qualquer computador com acesso à Internet através de um browser. De seguida encontram-se as funcionalidades da aplicação web com uma descrição.

- **Registar**

Permite ao utilizador registar uma nova conta de utilizador, para se poder autenticar.

- **Fazer login**

Permite ao utilizador autenticar-se, e consequentemente ter acesso às funcionalidades da aplicação.

- **Fazer logout**

Permite ao utilizador sair da sua conta.

- **Visualizar informações**

Permite ao utilizador ver informações sobre a aplicação e as técnicas usadas para a classificação de documentos.

- **Inserir dataset (Inserir conjunto de documentos de treino)**

Permite ao utilizador realizar o *upload* de um conjunto de treino (coleção de documentos), que deve estar dividido em pelo menos duas classes. O utilizador deve assim adicionar as classes, e inserir os respetivos documentos em cada uma destas. Deve ainda ser indicado se é pretendido um *dataset* privado (apenas o próprio utilizador pode visualizar e aplicar classificadores personalizados neste *dataset*) ou público (pode ser visualizado e utilizado para ser treinado através de classificadores, por qualquer utilizador registado no sistema).

O *upload* deve ser realizado através de um ficheiro em formato ".zip", contendo uma pasta com o nome do *dataset*. A pasta com o nome do *dataset* deve conter pelo duas diretorias correspondentes às classes do *dataset*. Os ficheiros de cada classe são inseridos nas suas respetivas diretorias em formato ".txt". A figura seguinte apresenta a composição necessária do ficheiro de *upload*.

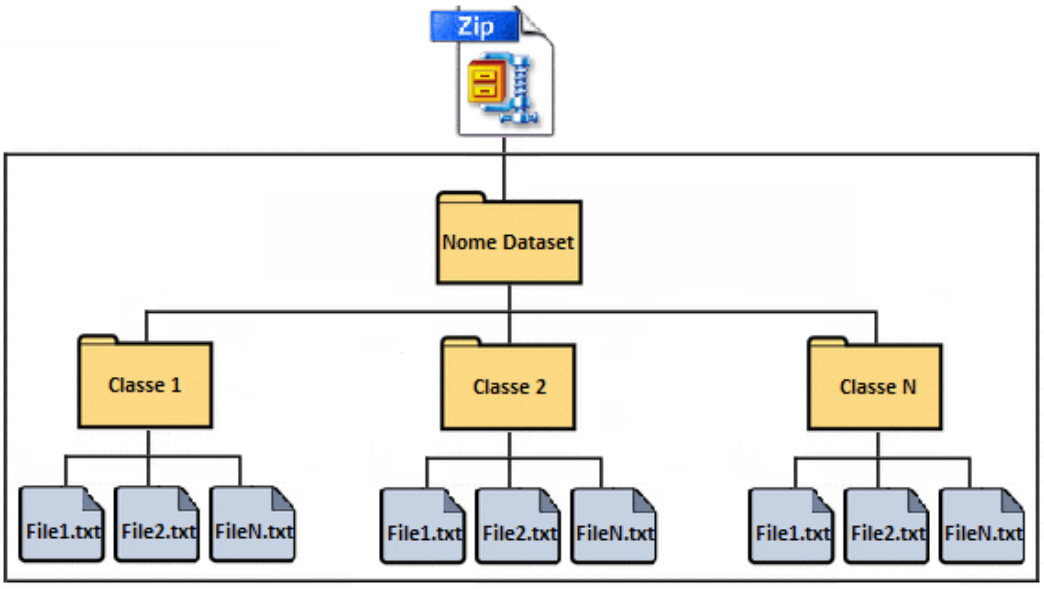


Figura 12 – Composição do ficheiro para upload do dataset

- **Remover dataset (Remover conjunto de documentos de treino)**

Permite ao utilizador remover um *dataset*, desde que tenha sido o próprio a adicioná-lo. Todos os classificadores, e os seus resultados são igualmente eliminados. Caso este seja público, mais nenhum utilizador poderá utilizar o mesmo para classificação de documentos.

- **Visualizar datasets privados**

Permite ao utilizador visualizar os *datasets* carregados pelo próprio assim como informações sobre o mesmo. A partir dos *datasets*, pode realizar outras funcionalidades, como remover o *dataset*, visualizar todos os classificadores criados pelo próprio a partir deste *dataset*, e configurar um novo classificador.

- **Visualizar datasets públicos**

Permite ao utilizador visualizar os *datasets* públicos e informações sobre o mesmo. A partir dos *datasets*, pode realizar outras funcionalidades, como visualizar todos os classificadores criados pelo próprio a partir deste *dataset*, e configurar um novo classificador.

- **Visualizar classificadores**

Permite ao utilizador visualizar os classificadores criados e treinados pelo próprio a partir de um determinado *dataset*. Os classificadores são apresentados apresentando as técnicas que este utiliza e a sua qualidade (*accuracy*, *recall*, *precision*). O utilizador nesta fase pode realizar várias operações a partir de um destes classificadores: pode ver os resultados (todos resultados de documentos classificados com este classificador), classificar novos documentos com este classificador, e eliminar o classificador.

- **Configurar novo classificador**

Permite ao utilizador configurar um classificador que pretenda, após ter escolhido o respetivo conjunto de treino. O configurador permite escolher os operadores de pré-processamento, escolher o classificador pretendido, e escolher o método de classificação.

- **Treinar classificador**

Permite ao utilizador treinar um classificador configurado por si, utilizando as técnicas previamente definidas.

- **Remover classificador**

Permite ao utilizador remover um classificador, desde que tenha sido o próprio a adicioná-lo. Todos os resultados de classificação de documentos deste classificador são igualmente eliminados.

- **Classificar documento(s)**

Permite ao utilizador classificar um ou vários documentos numa determinada classe (duas ou mais) definida num *dataset* a partir de um classificador previamente treinado. Para tal, o utilizador seleciona o classificador desejado, e faz o *upload* dos ficheiros desejados. O *upload* deve ser realizado através de um ficheiro em formato ".zip" contendo ficheiros em formato ".txt".

- **Visualizar resultados**

Permite ao utilizador visualizar todos os resultados da classificação de documentos a partir de um determinado classificador. Para cada um dos resultados, o utilizador tem a possibilidade de aceder ao resultado detalhado ou de remover o resultado.

- **Visualizar resultado detalhado**

Permite ao utilizador ver o resultado detalhado da classificação de um ou vários documentos previamente classificados. São apresentadas ao utilizador informações sobre o *dataset* e informações sobre o classificador. É ainda apresentado o número total de documentos classificados, assim como a quantidade de documentos classificados em cada uma das classes. Permite ainda ver a classificação de cada um dos ficheiros com a sua percentagem de certeza. A percentagem de documentos classificados em cada classe é apresentada num gráfico circular. O utilizador tem ainda a possibilidade de exportar o resultado da classificação em formato ".pdf".

- **Remover resultado**

Permite ao utilizador remover um resultado de classificação (conjunto de ficheiros classificados).

- **Exportar Resultados**

Permite ao utilizador exportar os resultados para um ficheiro em formato ".pdf". Neste ficheiro estão presentes detalhes sobre o *dataset*, sobre o classificador, sobre cada um dos ficheiros classificados e um gráfico com a percentagem dos ficheiros classificados em cada uma das classes.

3.3 Requisitos Não-Funcionais

Os requisitos não funcionais da aplicação são os seguintes:

- **Disponibilidade**
  - O servidor onde está alojada a aplicação web deve estar disponível 99% do tempo. Se houver necessidade de interromper o servidor para atualizações, deverá ser feito em horário noturno para causar o mínimo de transtorno aos utilizadores.

- **Performance**.
  - A comunicação da aplicação web com a *framework* http Bottle deve ser realizada em menos de 4 segundos.

- O sistema deve possibilitar a utilização em simultâneo de 20 utilizadores com perda de desempenho, no máximo de 10% em qualquer operação.

- **Robustez**

  - Os dados de entrada de formulários, exceto os de registo e login, devem ter valores *default* e devem ser usados sempre que o utilizador não preencha ou altere os campos.

- **Segurança**

  - Os utilizadores devem estar autenticados para aceder às funcionalidades de classificação da aplicação.

  - As *passwords* dos utilizadores devem serem encriptadas com a função de dispersão criptográfica MD5.

- **Usabilidade**

  - O interface do utilizador deve ser intuitivo e simples de usar.

  - O sistema deve apresentar uma barra de progresso quando está a ser treinado um modelo de classificação ou quando são classificados documentos.

  - O sistema deve apresentar mensagens de erro claras sempre que necessário.

  - O sistema deve poder ser acedido a partir de qualquer *browser* de qualquer sistema operacional.

3.4 Arquitetura do Sistema

Por arquitetura do sistema entende-se todos os componentes e respetivas interações entre eles que permitem atingir o objetivo do sistema.

Assim sendo, a arquitetura de alto nível do sistema pode ser representado de acordo com o seguinte esquema:

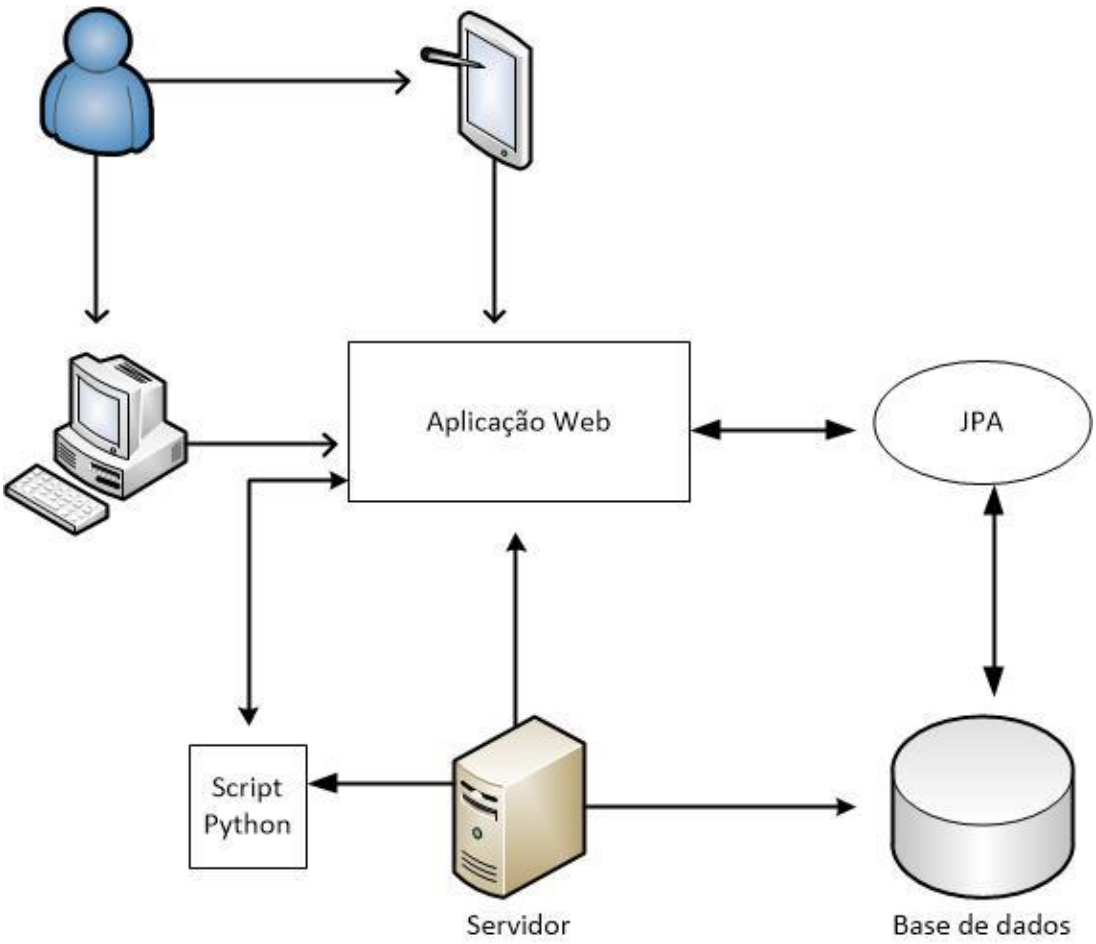


Figura 13 - Arquitetura do Sistema

Como se pode observar na figura acima, existem vários componentes na arquitetura do sistema:

- **Servidor** – Servidor onde se encontra o servidor de aplicação Glassfish para a plataforma J2EE suportando a aplicação web. Suporta também a base de dados com o servidor de base de dados MySQL e os scripts em Python para classificação.
- **Aplicação web** – Aplicação web (Capítulo 5) com que os utilizadores interagem através de um *browser*.
- **JPA (ORM)** – Mapeamento objeto relacional, que permite conectar a base de dados relacional com a programação orientada a objetos.
- **Base de dados** – Representam as estrutura de dados da plataforma (Capítulo 5.4).
- **Script python** – Scripts para classificação de documentos.

4 Casos de Estudo

4.1 Ferramentas utilizadas

De seguida é apresentado o *software* utilizado nos casos de estudos realizados.

4.1.1 Python(x,y)

Os scripts utilizados nos casos de estudo e na aplicação web para classificação de documentos, foram implementados com a ferramenta Python(x,y).

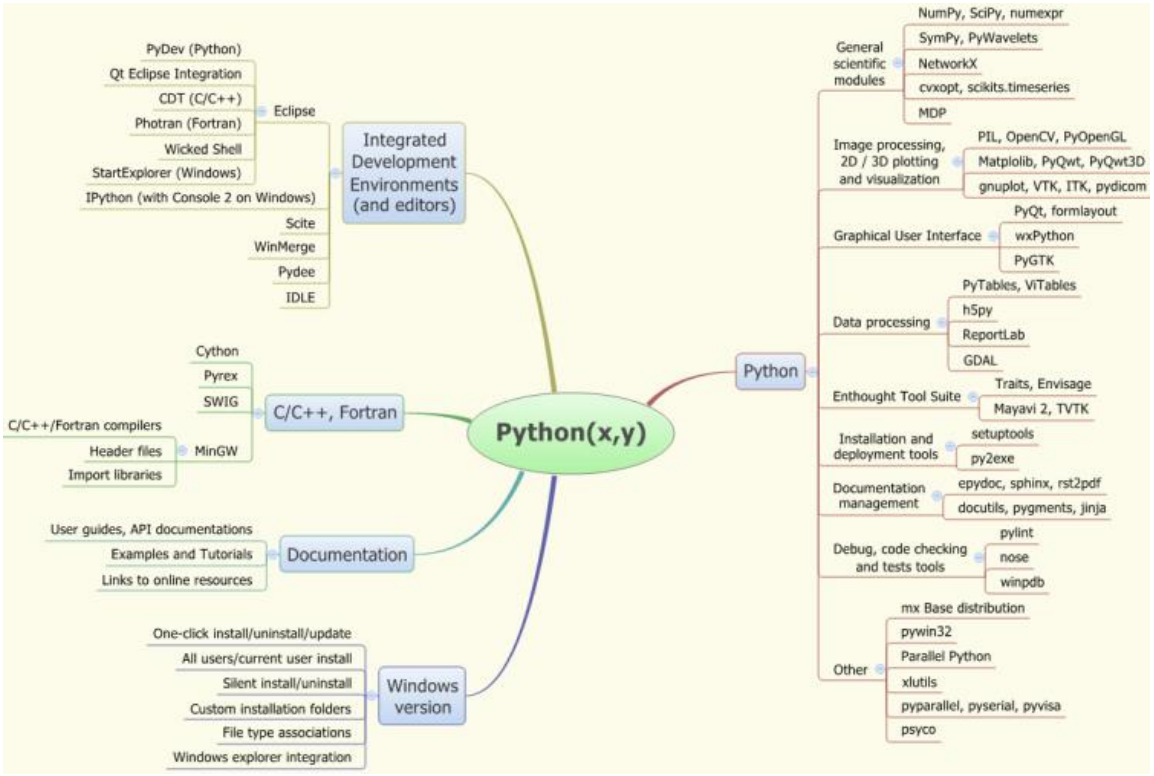
O desenvolvimento de aplicações usadas em ambiente científico é uma tarefa complexa, e para tal, os programadores envolvidos nestes projetos agradecem toda a ajuda para facilitar as suas tarefas. O Python(x,y) é precisamente um software adaptado para este tipo de tarefas, evitando aos programadores utilizar e testar todas as bibliotecas separadamente e procurar toda a documentação sobre as mesmas, o que levaria à perda de tempo.

O Python(x,y) é um software livre, de desenvolvimento científico e tecnológico para engenharia, desenvolvido para cálculos numéricos, análise de dados e visualização de dados em duas e três dimensões. A ferramenta tem base na linguagem de programação Python, com *framework* de desenvolvimento Qt, e com o ambiente de desenvolvimento integrado (IDE) Spyder. O Spyder é caracterizado como sendo intuitivo, interativo, e com semelhanças ao Matlab.

Este *software* permite a programação funcional simples com sintaxe semelhante ao Matlab, bem como programação orientada a objetos. Permite ainda o desenvolvimento de projetos científicos desde simples scripts até aplicações sofisticadas graças à estrutura de desenvolvimento Qt e o IDE Spyder com computação paralela em vários núcleos/processadores e *clusters*.

Um dos objetivos desta ferramenta é ajudar os programadores científicos que utilizam linguagens interpretadas, como Matlab ou IDL, ou linguagens compiladas, como C/C++ ou Fortran, a mudar para Python. Os programadores destas linguagens podem reutilizar os seus códigos sem alteração dos mesmos, podendo ser invocado diretamente a partir de scripts em Python. O inverso também se verifica podendo

scripts serem invocados a partir de código de outras linguagens, nomeadamente de Java através de Jython [34].



Fonte: https://python-xy.github.io/

Figura 14 - Python(x,y)

A linguagem Python é utilizada por grandes empresas, nomeadamente a Google e a NASA, esta última que refere: "*We chose Python because it provides maximum productivity, code that's clear and easy to maintain, strong and extensive (and growing!) libraries, and excellent capabilities for integration with other applications on any platform*" [35].

4.1.1.1 NLTK

O NLTK (Natural Language ToolKit) é uma biblioteca Python para processamento de linguagem natural e análise de texto. Inicialmente, foi concebida para o ensino, tendo sido adotada igualmente na área da investigação a desenvolvimento devido à sua utilidade e amplitude de cobertura. Assim, o NLTK destina-se à investigação e ao ensino de NLP e áreas tais que ciência cognitiva, inteligência artificial, recuperação de informação e *machine learning*. A biblioteca

NLTK define uma infraestrutura para construir programas de NLP em Python, fornecendo: classes básicas para representar dados relevantes ao processamento de linguagem natural; interfaces standard para realizar tarefas tais como *part-of-speech tagging*, análise sintática, classificação de texto, interpretação da semântica, *stemming*; implementações standard para cada tarefa que podem ser combinadas entre si para resolver problemas complexos [36].

O NLTK disponibiliza uma documentação extensa, em que o *website* http://www.nltk.org/ fornece toda a documentação da API para cada módulo, classes e funções do *toolkit*, especificando os parâmetros necessários e oferecendo exemplos de utilização. A biblioteca disponibiliza igualmente um fórum de discussão ativo em https://groups.google.com/forum/#!forum/nltk-users. Um livro acompanha a ferramenta, intitulado de *Natural Language Processing with Python*, este explica e exemplifica os conceitos e funcionalidades do NLTK.

O NLTK foi desenhado com quatro grandes objetivos: simplicidade, consistência, extensibilidade e modularidade. Os módulos do NLTK estão representados na seguinte tabela [24]:

Tarefa de NLP	Módulo NLTK
Acesso ao corpora	nltk.corpus
Processamento de Strings	nltk.tokenize, ntkl.stem
Collocation discovery	nltk.collocations
Part-of-speech tagging	nltk.tag
Classificação	nltk.classify, nltk.cluster
Chunking (n-grams)	nltk.chunk
Parsing	nltk.parse
Interpretação da semântica	nltk.sem, nltk.inference
Métricas de avaliação	nltk.metrics
Probabilidade e estimativa	nltk.probability

Aplicações	nltk.app, nltk.chat
Trabalho de campo linguístico	nltk.toolbox

Tabela 15 – Módulos do NLTK

4.1.1.2 Scikit-learn (Sklearn)

O Scikit-learn (sklearn) é um *framework open-source* de *machine learning* para Python acessível para todos, e reutilizável em vários contextos. Atualmente é considerada uma das melhores bibliotecas e mais eficientes para aplicações de *data mining* e de análise de dados. Esta biblioteca oferece um amplo conjunto de algoritmos de *machine learning* supervisionados e não-supervisionado para as mais diversas finalidades, a partir de uma interface consistente e de fácil usabilidade.

O Scikit-learn é baseado num conjunto de outras bibliotecas:

- **NumPy –** Pacote para manipulação de *arrays* N-dimensional.
- **SciPy –** Pacote para computação científica.
- **Matplotlib** – Pacote para gráficos 2D e 3D.
- **IPython** – Pacote para terminal interativo de comandos.
- **Sympy** – Pacote para matemática simbólica.
- **Pandas** – Pacote para análise de estruturas de dados.

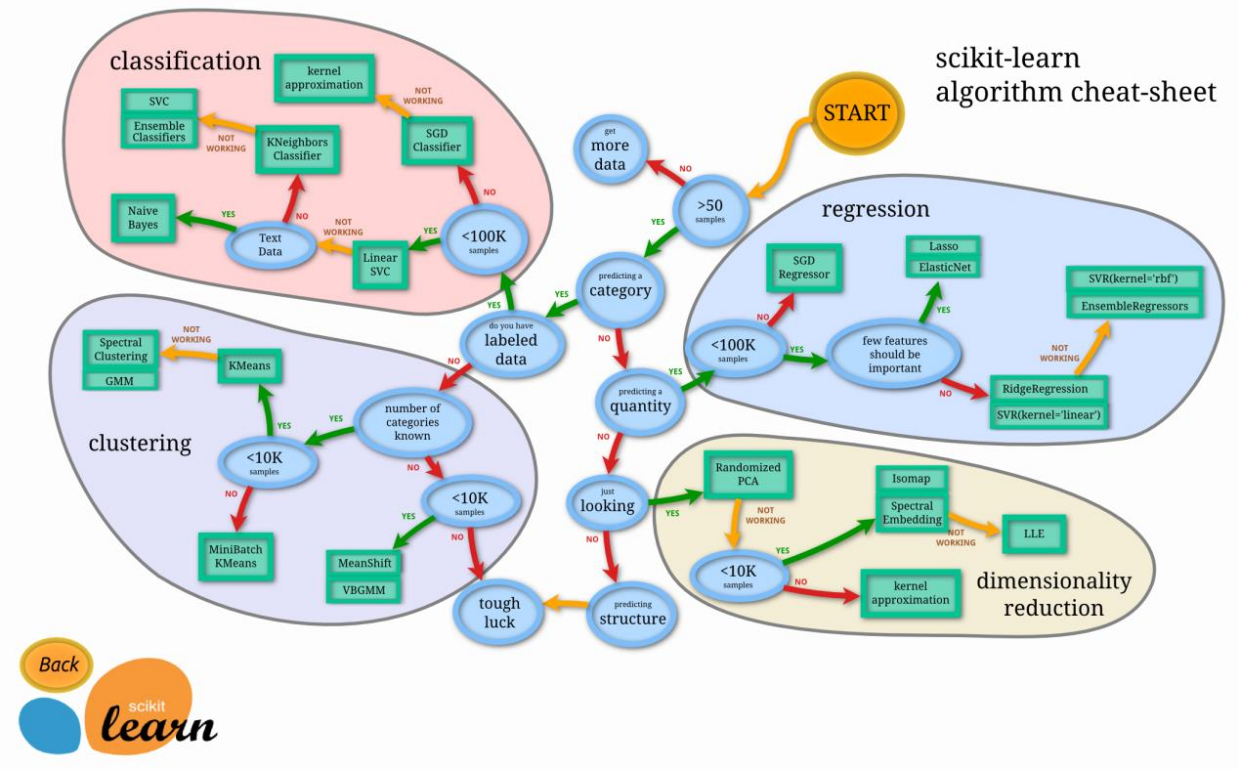
A biblioteca para além de oferecer suporte e robustez para ambientes de produção, a biblioteca Scikit-learn também se foca em usabilidade, limpeza de código, colaboração, bastante documentação e um ótimo desempenho. As funções disponibilizadas dispõem das facilidades da linguagem Python, mas possuem mecanismos implementados em linguagens de baixo nível, como C, de forma a otimizar ao máximo a utilização de recursos de *hardware*. O Scikit-learn é utilizado por grandes empresas e dispõe de uma grande comunidade.

As principais funcionalidades oferecidas pelo Scikit-learn são:

- **Clustering** - Algoritmos não-supervisionados para agrupamento de dados não categorizados, como *K-means*.

- **Cross Validation** - Técnica para validação de modelos preditivos de algoritmos supervisionados contra dados não vistos.
- **Datasets** - Oferece um conjunto de bases de dados para aplicação das diferentes técnicas de aprendizado.
- **Dimensionality Reduction** - Para redução do número de atributos a serem analisados de forma a otimizar a análise.
- **Ensemble methods** - Para combinação das predições de diversos modelos
- **Feature extraction** - Para definição de atributos em dados não estruturados como imagem e texto.
- **Feature selection** - Para identificação de atributos estatisticamente relevantes para criação de modelos preditivos supervisionados.
- **Parameter Tuning** - Para otimização de execução de modelos preditivos supervisionados.
- **Manifold Learning** - Para análise descritiva de dados complexos e multidimensionais.

Uma das maiores dificuldades na resolução de problemas de *machine learning* pode ser encontrar ou escolher o modelo mais apropriado para o problema. Diferentes modelos podem ser melhores para diferentes tipos de dados e diferentes problemas. A seguinte figura mostra algumas funcionalidades e modelos preditivos supervisionados e não supervisionados, como guia sobre a forma de abordar os problemas corretamente, disponibilizado na página web da biblioteca [37].



Fonte: http://scikit-learn.org/stable/tutorial/machine_learning_map/

Figura 15 – Modelos Scikit-learn

4.2 Casos de Estudo e Resultados

Na implementação dos scripts para classificação, e de forma a testar, avaliar e melhorar os resultados foram utilizados três conjuntos de documentos com texto natural e sem qualquer tratamento prévio dos textos, o *dataset* Movies Reviews, o *dataset* Ohsumed e um *dataset* de Enzimas.

Nos casos de estudos foram implementados duas versões de código em Python para classificação de documentos. A segunda versão apresenta melhorias e novas funcionalidades em relação à primeira, resultando em melhores resultados.

- **Primeira versão do código**

  A primeira versão do código para classificação de documentos foi implementada em Python com base no manual *Natural Language Processing with Python*, através da biblioteca NLTK e foi testada através do *dataset* Movie Reviews.

Foram criadas várias funções de pré-processamento de texto: transform cases (lower case) para transformar os termos em letra minúscula, tokenize (*non letters*) de forma a criar os *tokens* e eliminar *tokens* que não sejam palavras como pontuações, *stemming* (*snowball*) e *prune* (percentual).

Na criação de vetores foram usados dois métodos, o TF-IDF e o binário. Neste estudo foram apenas considerados as 2000 palavras mais frequentes em todo conjunto de documentos.

A divisão dos conjuntos para treino e para teste, foram testados com o *cross-validation* dividido em quatro *subsets*, e pelo método de percentagem em que 70% dos documentos positivos e 70% dos documentos negativos foram usados para treino e os restantes 30% de documentos de cada classe foram usados para treino.

Os classificadores usados pertencem ao NLTK e ao Scikit-Learn. Foram usados os seguintes classificadores do NLTK: *Decision Tree*, *Maximum Entropy* (algoritmo iis e algoritmo gis). Os classificadores utilizados provenientes do Scikit-Learn foram estes: *Logistic Regression*, NB Bernoulli, NB Multinomial, Linear SVC, Nu SVC, e SVC.

- **Segunda versão do código**

  A segunda versão do código para classificação de documentos foi implementada em Python e testada através dos três *datasets*: Movie Reviews, Ohsumed; *dataset* das Enzimas. Esta versão da implementação é a versão utilizada pela aplicação web e permite a classificação de documentos em duas ou mais classes.

  Quando um *dataset* tem mais de 2 classes, em que *x* é o número de classes, são realizados *x* classificadores. Cada classificador tem documentos da própria classe (classe positiva) para treino e teste, e amostras das restantes *x-1* classes para treino e teste, preservando a percentagem de amostras para cada classe.

  Foram criadas várias funções de pré-processamento de texto: *Transform Cases* (*lower case*) para transformar os termos em letra minúscula, remoção de *Non-*

Letters, remoção de *stopwords*, e *stemming* (*Snowball*). Estas funções foram realizadas com recurso a métodos disponibilizados pelo NLTK.

Na criação de vetores foram usados dois métodos, o TF-IDF e o binário. Após o pré-processamento do texto, é possível utilizar: todas as palavras para treino; utilizar o número desejado de palavras com maior peso de classificação das classes (melhores *features*); combinar as melhores *features* individuais do ponto anterior com as melhores *features* de *bigrams*.

Para obter as palavras com mais ganho de informação foi utilizado a técnica *Chi Square*. O NLTK fornece a classe BigramAssocMeasures que contém um método para cálculo do ganho de informação de cada *feature*. Para tal, inicialmente foram calculadas algumas frequências de cada palavra, a sua frequência global e a sua frequência dentro de cada classe. Após a obtenção das frequências é possível obter o valor de ganho de informação através da função BigramAssocMeasures.chi_sq. Todos os termos são colocados numa lista com o seu respetivo valor de ganho de informação, e o utilizador pode então escolher o número de melhores *features* a partir do topo da lista. O mesmo processo é realizado com os *bigrams*.

O script para classificação permite a divisão dos conjuntos para treino e para teste, com a técnica *stratified* cross-validation, com o *stratified cross-validation* aleatório ou com percentagem 70% dos documentos para treino e 30% dos documentos para teste. O *stratified cross-validation* foi realizado com recurso à função cross_validation.StratifiedKFold disponibilizada pelo Scikit-Learn.

Os classificadores usados pertencem ao NLTK e ao Scikit-Learn. Foi usado o seguinte classificador do NLTK: *Naive Bayes*. Os classificadores utilizados provenientes do Scikit-Learn foram estes: *Logistic Regression*, NB Bernoulli, NB Multinomial, Linear SVC, Nu SVC, e SVC.

A avaliação dos classificadores é obtida através de métodos do NLTK, que disponibiliza as métricas *recall*, *precision* e *accuracy*.

Para treinar um modelo de classificação, o Bottle (Capítulo 5.3) invoca os métodos implementados neste script pela seguinte ordem.

1. classificador = **Classificador("None")**

   - Criação de uma instância da classe Classificador.

2. classificador.**definirIDClassificador(idClassificador)**

   - Define um ID ao classificador (o mesmo que é atribuído na base de dados).

3. classificador.**definirDataset(paramNomeDataset)**

   - Através do nome do *dataset* proveniente do argumento *paramNomeDataset* os ficheiros são carregados para memória. Esta função define as classes do *dataset* através da estrutura da diretoria do *dataset*.

4. classificador.**definirTipoTreinoTeste(paramTipoTreinoTeste)**

   - Define o tipo de treino e teste a usar pelo classificador (70%/30%;*cross-validation;cross-validation* aleatório).

5. classificador.**definirOperadores(op1,op2,op3,op4)**

   - Define os operadores de pré-processamento a usar pelo classificador. (remoção de *non letters*; remoção de *stopwords*; transformação para letras minúsculas; *stemming*).

6. classificador.**tipoClassificador(paramTipoClassificador)**

   - Define o classificador que será usado (NaiveBayesClassifier; MultinomialNB; BernoulliNB; NuSVC; LinearSVC; LogisticRegression).

7. classificador.**definirMetodoCaracterizacao(paramMelhoresPalavras,paramMelhoresNGrams)**

   - Define a representação do *bag of words(Binário ou TF-IDF)* e da seleção de melhores *features* (Número de melhores *features* e número de melhores *features bigrams*).

8. classificador.**treinar()**

- Treina o modelo de classificação com todos os parâmetros definidos pelos métodos invocados anteriores. Nesta fase são aplicadas todas as técnicas de pré-processamento, criação dos vetores, seleção das *features* e treino. Este método invoca de sua vez, muitos outros métodos implementados a fim de realizar todas operações.

9. retorno = classificador.**getAvaliacao()**

   - Calcula e retorna a avaliação do modelo treinado. O retorno é constituído pelas métricas de *accuracy*, *precision* e *recall*.

Para classificar documentos a partir de um classificador criado anteriormente, o Bottle (Capítulo 5.3) invoca o seguinte método.

1. retorno = **classificarDocumentos(pastaFicheiros)**

   - Classifica e retorna a classe e a percentagem de certeza de cada ficheiro classificado.

De seguida são apresentados os *dataset* utilizados, e a análise dos resultados obtidos nos diferentes estudos realizados. Todos os estudos foram realizados com as métricas de avaliação *accuracy*, *recall* e *precision*. Os gráficos presentes neste Capítulo mostram os resultados com a métrica mais importante, o *accuracy*.

4.2.1 Movie Reviews

Este conjunto de documentos é relativo à crítica de filmes e é composto por 2000 documentos divididos em duas classes: crítica positiva (1000 documentos) e crítica negativa (1000 documentos) [38].

4.2.1.1 Movie Reviews - Resultados com 1ª versão do código

Todos os resultados deste estudo estão disponíveis em anexo (Anexo E - Resultados de classificação de documentos com Python – 1ª versão (Movie Reviews)).

Neste estudo foi utilizado o *dataset* Movie Reviews com *cross-validation* de 4 *subsets* e divisão de 70% para treino e 30% para testes. Visto se tratar dum *dataset* com bastantes documentos, os resultados 70% para treino e 30% para teste são próximos dos resultados com *cross-validation*, como tal, são aqui apresentados os resultados com o método da divisão dos conjuntos por percentagem. Os resultados com *cross-validation* podem ser consultados no anexo E. Neste estudo foram selecionadas as 2000 *features* mais frequentes no Corpus.

Os classificadores utilizados neste estudo foram todos aqueles disponibilizados pelo NLTK e Scikit-Learn: *Decision Tree*, *Maximum Entropy*, *Logistic Regression*, NB Bernoulli, NB Multinomial, Linear SVC, Nu SVC e SVC.

De seguida é apresentado um gráfico com os resultados com a representação dos vetores binários.

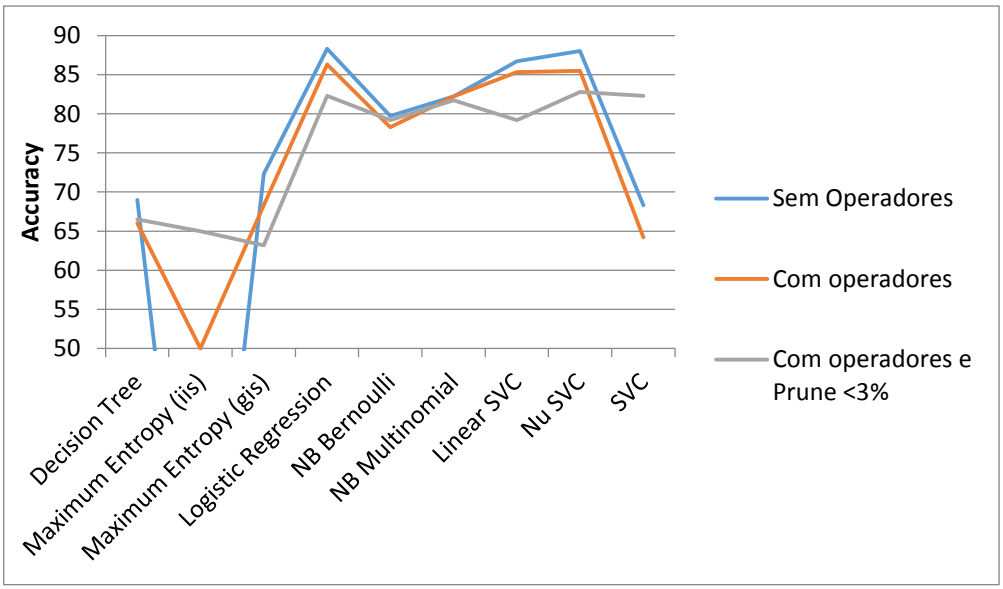


Figura 16 – Resultados 1ªa versão – Binário

O classificador *Maximum Entropy* (iis) deu um erro de memória no teste sem operadores. Neste teste a primeira conclusão foi que o *Decision Tree* e os dois algoritmos de *Maximum Entropy* não apresentaram resultados satisfatórios e não seriam usados na segunda versão do código. O classificador *Logistic Regression* apresentou excelente resultados atingindo os 88.3% de *accuracy* sem operadores, tal como o Nu SVC com 88%. Os classificadores de *Naive Bayes* apresentaram resultados entre 80 e 85%. O classificador SVC foi o único que apresentou resultados bastantes superiores com operadores e *Prune*<3%. Os resultados inferiores com

Prune<3% em relação aos outros resultados, podem ser explicados pelo facto de poderem estar a ser eliminadas *features* com bastante peso para definir classes.

De seguida é apresentado um gráfico com os resultados com a representação dos vetores com TF-IDF.

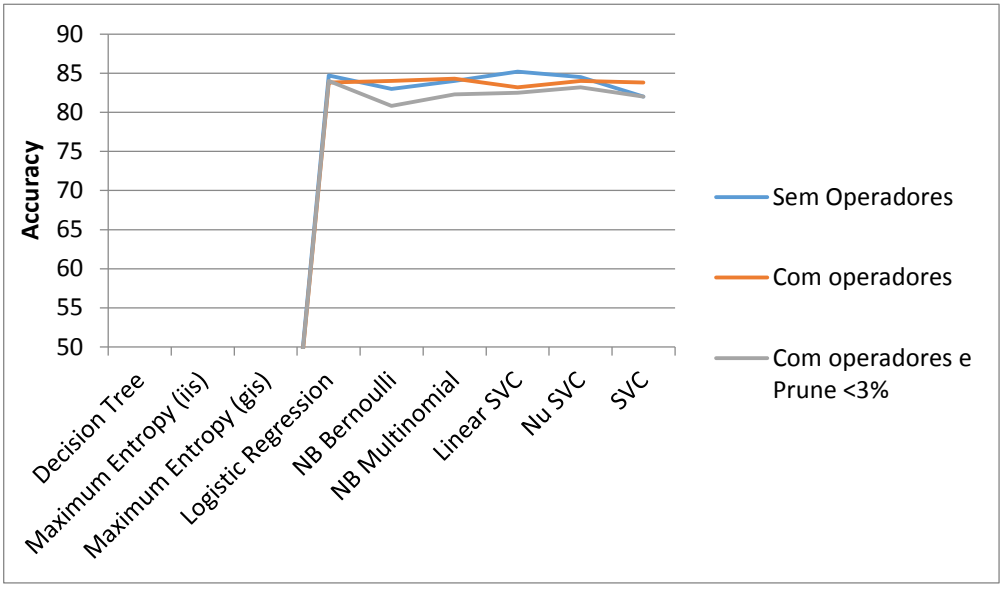


Figura 17 – Resultados 1ªa versão – TF-IDF

Com a representação dos vetores TF-IDF, os classificadores que apresentaram piores resultados com a representação binária, o *Decision Tree* e os dois algoritmos *Maximum Entropy*, deram erro de memória.

Apesar dos máximos do *Logistic Regression* e do Nu SVC terem baixado os valores em relação à representação binária, os outros classificadores no geral subiram os resultados para valores próximos dos 85%. Pode ser verificar também que os resultados obtidos foram muito mais uniformes com o TF-IDF.

Com este estudo, concluiu-se que o *Decision Tree*, e os algoritmos *Maximum Entropy*, ao apresentarem resultados baixos e ao darem erro de memória, não seriam usados na segunda versão do código.

4.2.1.2 Movie Reviews - Resultados com 2ª versão do código

De seguida são apresentados os principais resultados obtidos com o *dataset* Movie Reviews com 70% dos documentos para treino e 30% para teste. Os restantes resultados nomeadamente com *cross-validation* estão no anexo F.

De seguida é apresentado um gráfico com os resultados com o classificador *Naive Bayes* do NLTK.

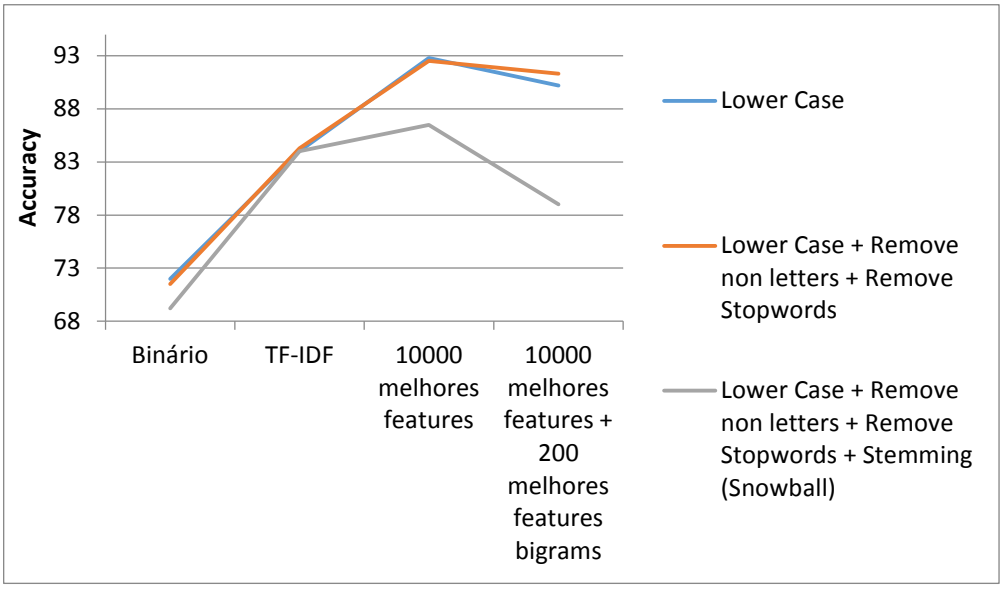


Figura 18 - Movie Reviews - Naive Bayes

O melhor resultado com as *Naive Bayes* foi de 92.8% de *accuracy* com a escolha das 10000 melhores *features*, com a transformação para letras mínusculas.

De seguida é apresentado um gráfico com os resultados com o classificador Multinomial NB do Scikit-Learn.

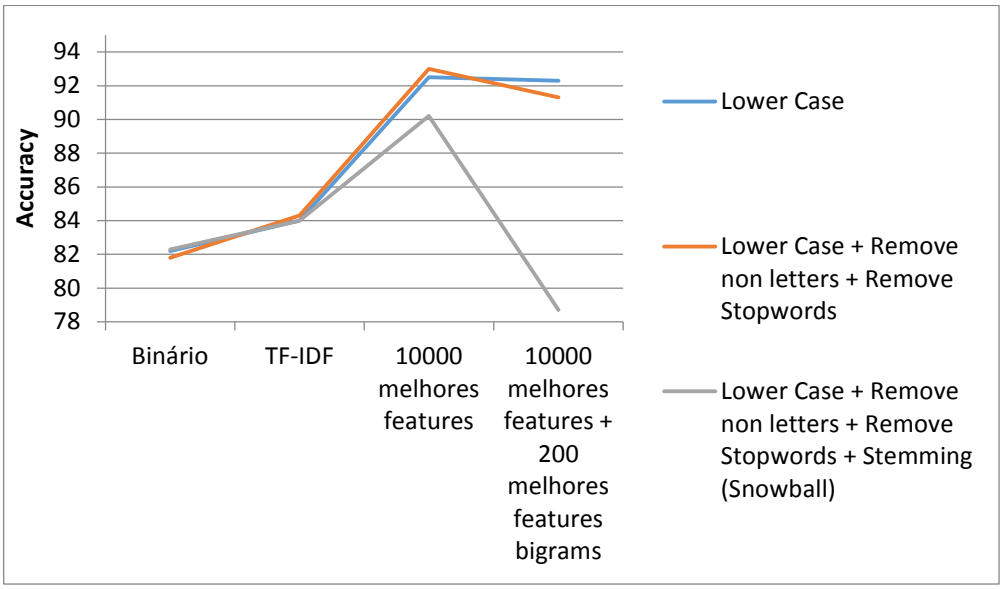


Figura 19 – Movie Reviews – Multinomial NB

O melhor resultado com o classificador Multinomial NB foi de 92.5% de *accuracy* com a escolha das 10000 melhores *features*, com a transformação para letras mínusculas.

De seguida é apresentado um gráfico com os resultados com o classificador Bernoulli NB do Scikit-Learn.

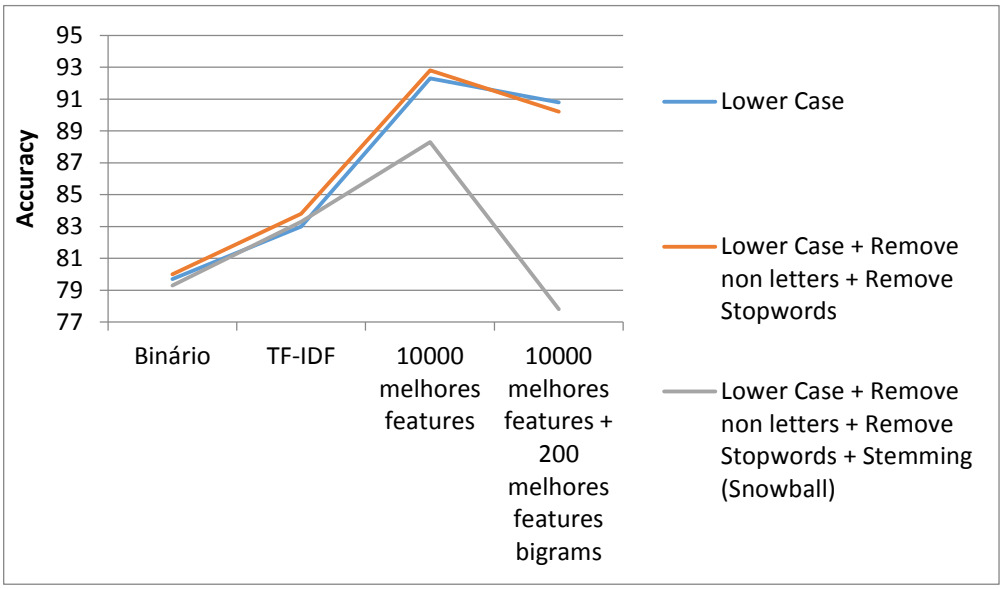


Figura 20 – Movie Reviews – Bernoulli NB

O melhor resultado com o classificador Bernoulli NB foi de 92.8% de *accuracy* com a escolha das 10000 melhores *features*, com a transformação para letras minúsculas e com remoção de *non letters* e *stopwords*.

De seguida é apresentado um gráfico com os resultados com o classificador Linear SVC do Scikit-Learn.

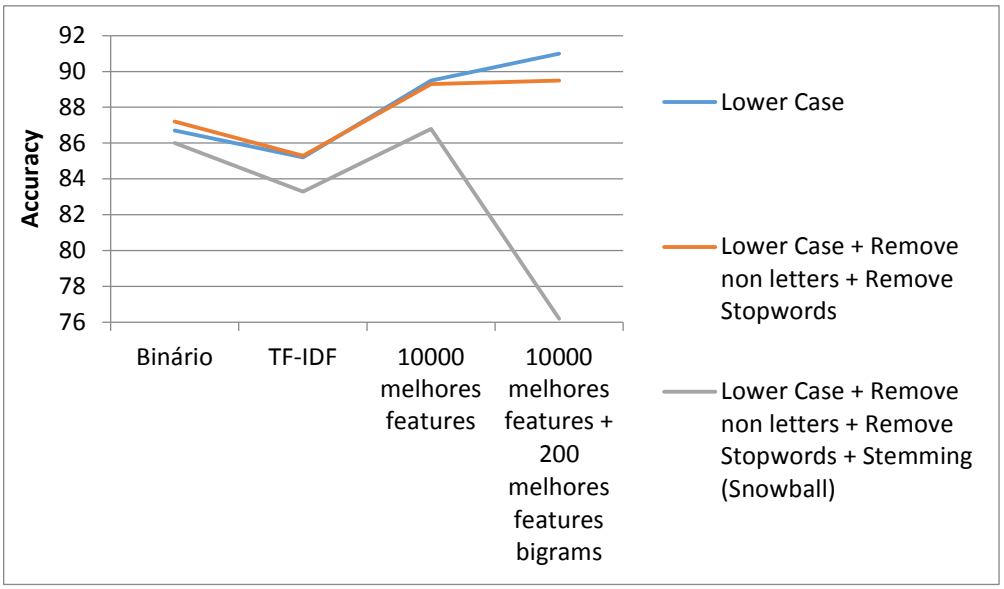


Figura 21 – Movie Reviews – Linear SVC

O melhor resultado com o classificador Linear SVC foi de 91% de *accuracy* com a escolha das 10000 melhores *features* e das 200 melhores *features bigrams*, com a transformação para letras minúsculas.

De seguida é apresentado um gráfico com os resultados com o classificador Nu SVC do Scikit-Learn.

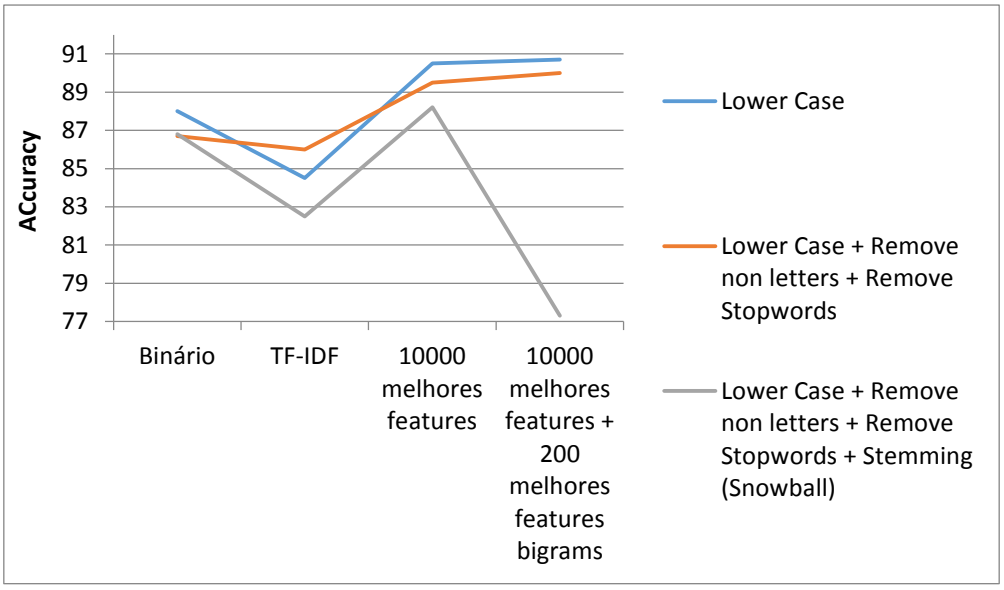


Figura 22 – Movie Reviews – Nu SVC

O melhor resultado com o classificador Nu SVC foi de 90.7% de *accuracy* com a escolha das 10000 melhores *features* e das 200 melhores *features bigrams*, com a transformação para letras minúsculas.

De seguida é apresentado um gráfico com os resultados com o classificador *Logistic Regression* do Scikit-Learn.

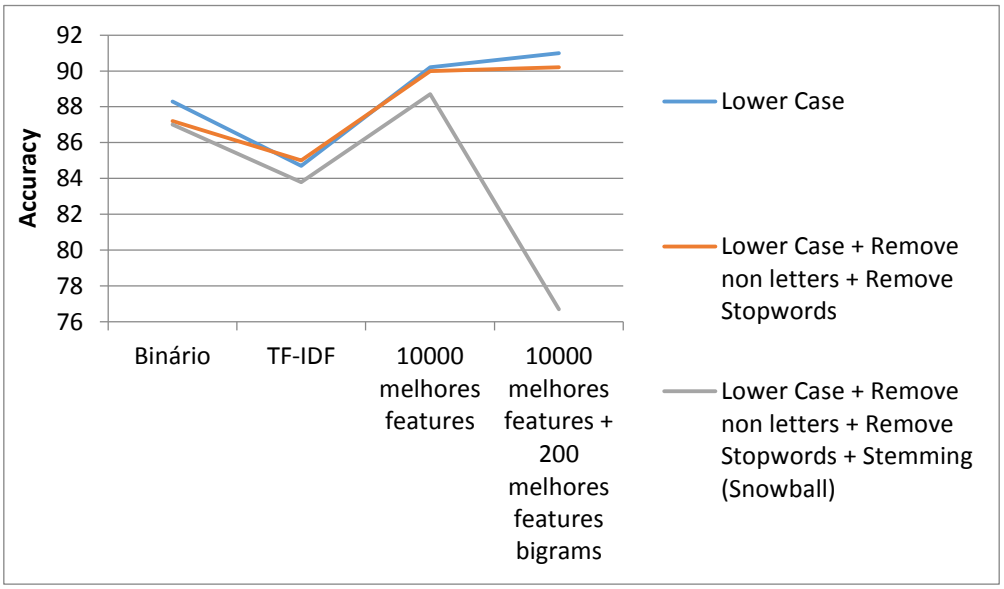


Figura 23 – Movie Reviews – Logistic Regression

O melhor resultado com o classificador *Logistic Regression* foi de 91% de *accuracy* com a escolha das 10000 melhores *features* e das 200 melhores *features bigrams*, com a transformação para letras minúsculas.

Nos resultados obtidos com os diferentes classificadores é possível verificar que a utilização das melhores *features* apresenta excelente resultados (acima dos 90% de *accuracy*) comparando com a utilização de todas as palavras com representação binária ou TF-IDF (entre 85% a 88% de *accuracy*). A redução da dimensão dos vetores com as melhores *features*, aumenta significativamente o desempenho e a qualidade dos modelos de classificação.

Conclui-se também que ao selecionar as melhores *features bigrams* juntamente com as melhores *features* e *stemming*, os resultados perdem bastante qualidade.

Os resultados com as melhores *features bigrams* apresentaram melhoria em relação à utilização exclusiva das melhores *features* em alguns classificadores (Linear

SVC, Nu SVC e *Logistic Regression*), enquanto noutros perderam qualidade (*Naive Bayes*, Multinomial NB, Bernoulli NB).

Os melhores resultados dos diferentes classificadores atingiram valores médios de 92.5%, sendo o melhor que o melhor resultado foi de 93% de *accuracy* pelo classificador Multinomial NB do Scikit-Learn, com operadores de pré-processamento (*lower* case, *remove non letters*, *remove stopwords*), e com a seleção das 10000 melhores *features*.

4.2.2 Ohsumed

Este conjunto de documentos é relativo a resumos de artigos e jornais de medicina e é composto por 56984 documentos divididos em vinte e três classes. A tabela seguinte mostra os detalhes das classes e o número de documentos presentes em cada uma das classes [39].

Classe	Descrição	Número de documentos
C01	Bacterial Infections and Mycoses	2540
C02	Virus Diseases	1171
C03	Parasitic Diseases	427
C04	Neoplasms	6327
C05	Musculoskeletal Diseases	1678
C06	Digestive System Diseases	2989
C07	Stomatognathic Diseases	526
C08	Respiratory Tract Diseases	2589
C09	Otorhinolaryngologic Diseases	715
C10	Nervous System Diseases	3851
C11	Eye Diseases	998
C12	Urologic and Male Genital Diseases	2518
C13	Female Genital Diseases	1623
C14	Cardiovascular Diseases	6102
C15	Hemic and Lymphatic Diseases	1277
C16	Neonatal Diseases and Abnormalities	1086
C17	Skin and Connective Tissue Diseases	1617
C18	Nutritional and Metabolic Diseases	1919

C19	Endocrine Diseases	865
C20	Immunologic Diseases	3116
C21	Disorders of Environmental Origin	2933
C22	Animal Diseases	506
C23	Pathological Conditions, Signs,	9611

Tabela 16 – Dataset Ohsumed

4.2.2.1 Ohsumed - Resultados com 2ª versão do código

De seguida são apresentados os principais resultados obtidos com o *dataset* Ohsumed. Este *dataset* tem a particularidade de ser um conjunto com mais de duas classes, neste caso vinte e três classes. Neste caso os resultados para cada classe foram obtidos utilizando 70% dos documentos da própria classe (classe positiva) para treino, e a classe negativa representa o mesmo número de documentos da classe positiva divididos por documentos das restantes 22 classes. Os 30% dos documentos de teste seguem o mesmo princípio do conjunto de treino.

Um dos principais objetivos ao utilizar este *dataset*, foi a possibilidade de testar o algoritmo com um *dataset* com mais de duas classes, e o facto de representar artigos da área de medicina. Para tal, e pelo facto dos classificadores já terem sido testados com o *dataset* Movie Reviews, os testes realizados foram apenas realizados com o classificador *Naive Bayes* com a vertente da seleção das melhores *features*, e sem operadores de pré-processamento com a exceção da transformação para letra minúscula.

De seguida é apresentado um gráfico com os resultados obtidos.

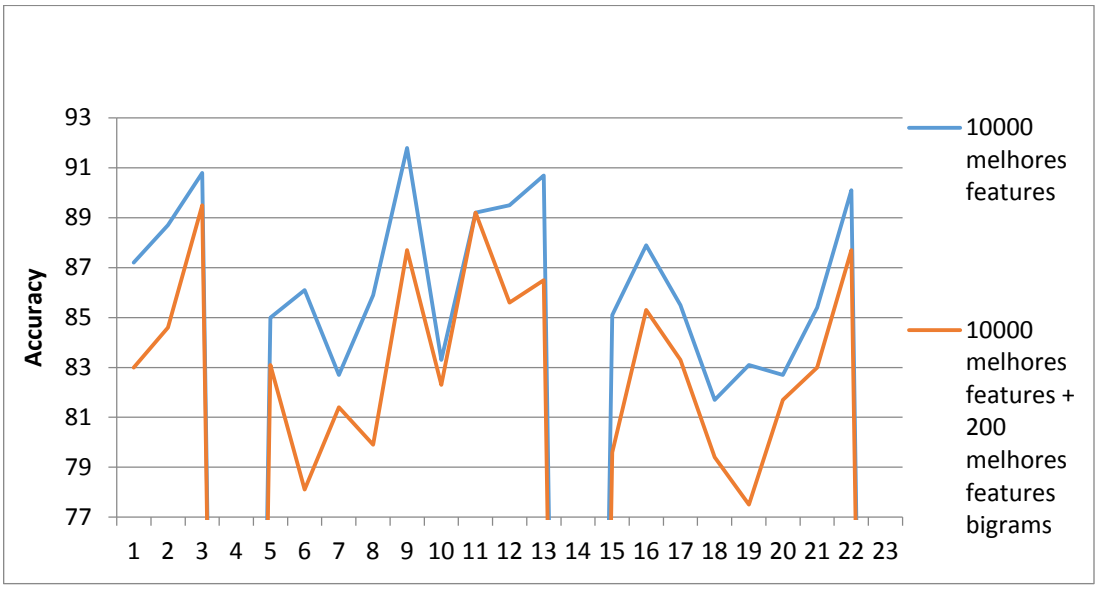


Figura 24 - Resultados com Ohsumed

Os testes com as classes C04, C14 e C23 deram erro de memória. Pode-se verificar que utilizando apenas as melhores *features* ao invés de utilizar também as melhores *features bigrams*, os resultados são melhores em qualquer uma das classes. Os resultados foram bastante bons na medida em que é um *dataset* com 23 classes, com um resultado médio de 86.62% de *accuracy* com as 10000 melhores *features*. O resultado médio com as 10000 melhores *features* e 200 melhores *features bigrams* foi de 83.42% de *accuracy*. Geralmente, quanto mais classes um conjunto de documentos tiver, mais os resultados tendem a baixar.

4.2.3 Enzimas

Este *dataset* foi criado manualmente e pretende discriminar duas subfamílias de enzimas, as pepsinas (*pepsin*) pertencentes à família de peptidases A1 e as quimotripsinas (*chymotrypsin*) pertencentes à família de peptidases S1. As pepsinas são enzimas digestivas com a função de desdobrar proteínas em péptidos mais simples (aminoácidos). As quimotripsinas são enzimas digestivas que têm como função clivar ligações peptídicas.

O conjunto de documentos foi obtido a partir do Merops, uma plataforma online que reúne literatura sobre peptidases. O website apresenta informações sobre todas as peptidases divididas em famílias e clans, assim como os links do PubMed para os artigos relacionados. A vantagem do Merops é que a literatura apresentada

sobre cada enzima já foi verificada e é possível assim, saber que os artigos científicos tratam realmente das enzimas em questão, resultando em dados de treino de grande qualidade [40].

Os documentos obtidos para treino são referentes aos resumos dos artigos, e foram obtidos a partir da plataforma PubMed, uma versão gratuita da base de dados Medline. O PubMed contem mais de 25 milhões de referências de literatura biomédica da Medline, e de livros online [41].

O *dataset* obtido contem 270 documentos com resumos de artigos, divididos em duas classes: pepsinas com 135 documentos e quimotripsinas com 135 documentos.

4.2.3.1 Enzimas - Resultados com 2ª versão do código

Neste subcapítulo são apresentados os principais resultados obtidos com o *dataset* das Enzimas com 70% dos documentos para treino e 30% para teste. Os resultados detalhados, nomeadamente com os valores de *recall* e *precision* estão em anexo (Anexo H – Resultados de classificação com Python – 2ª versão (Enzimas)).

De seguida é apresentado um gráfico com os resultados com o classificador *Naive Bayes* do NLTK.

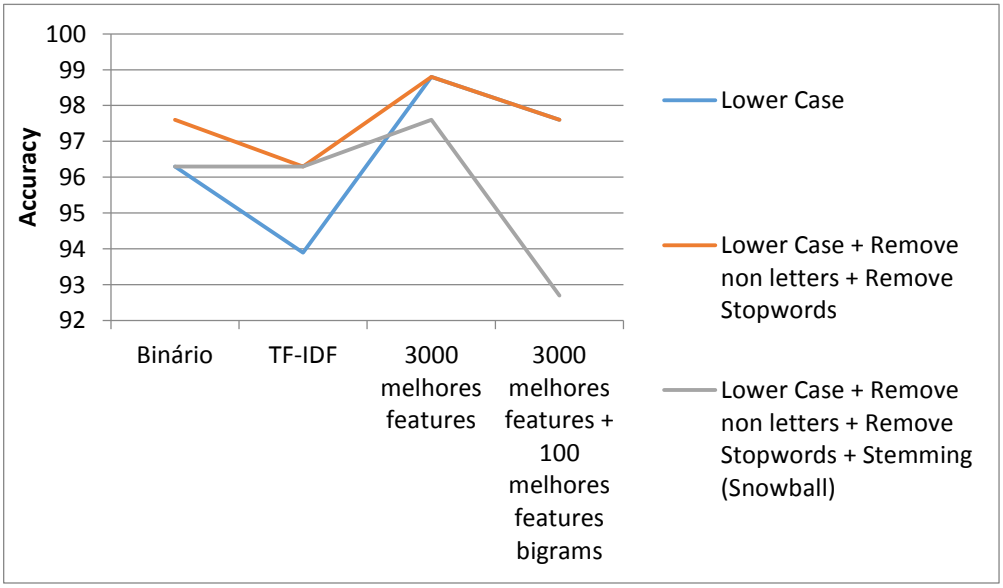


Figura 25 - Enzimas - Naive Bayes

O melhor resultado com as *Naive Bayes* foi de 98.8% de *accuracy* com a escolha das 3000 melhores *features*, com a transformação para letras minúsculas e com remoção de *non letters* e *stopwords*.

De seguida é apresentado um gráfico com os resultados com o classificador Multinomial NB do Scikit-Learn.

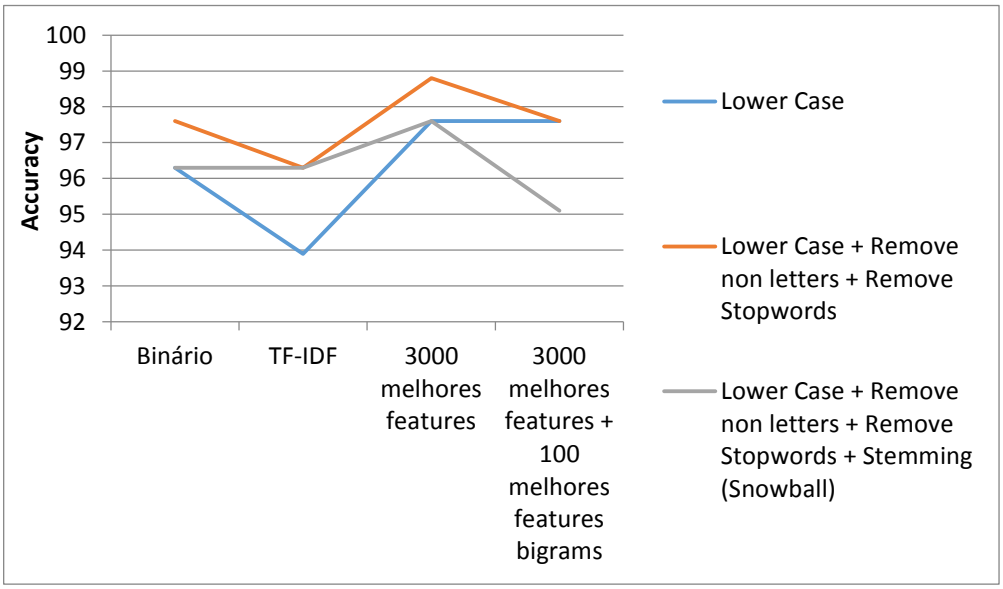


Figura 26 – Enzimas – Multinomial NB

O melhor resultado com o classificador Multinomial NB foi de 98.8% de *accuracy* com a escolha das 3000 melhores *features*, com a transformação para letras minúsculas e com remoção de *non letters* e *stopwords*.

De seguida é apresentado um gráfico com os resultados com o classificador Bernoulli NB do Scikit-Learn.

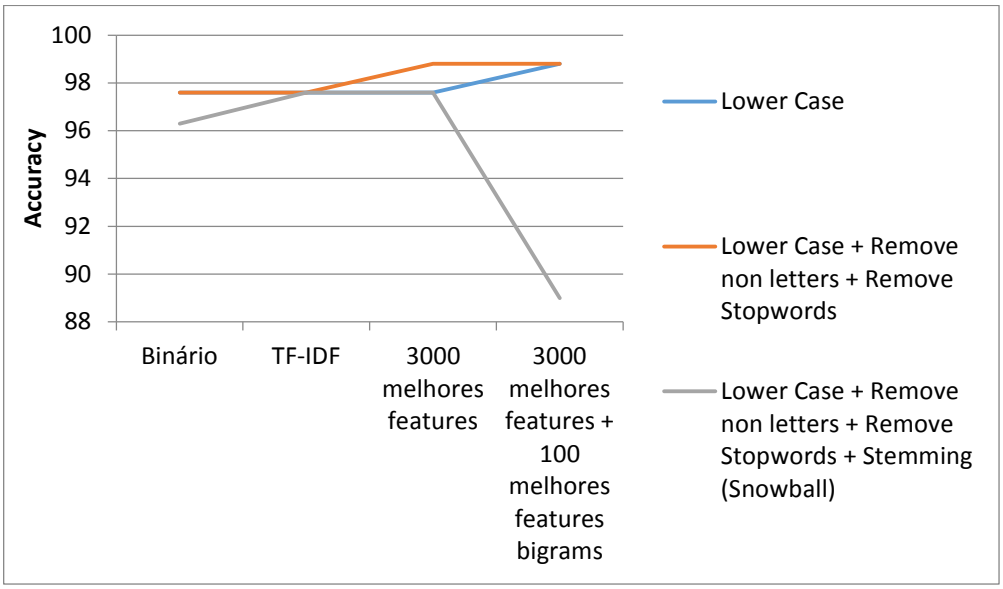


Figura 27 – Enzimas – Bernoulli NB

O melhor resultado com o classificador Bernoulli NB foi de 98.8% de *accuracy* com a escolha das 3000 melhores *features* e os operadores de pré-processamento de transformação para letras minúsculas, e remoção de *non letters* e *stopwords*. A seleção das melhores *features* em conjunto com os melhores *bigrams*, em conjunto com mesmos operadores de pré-processamento do resultado descrito anteriormente, obteve o mesmo resultado de 98.8%.

De seguida é apresentado um gráfico com os resultados com o classificador Linear SVC do Scikit-Learn.

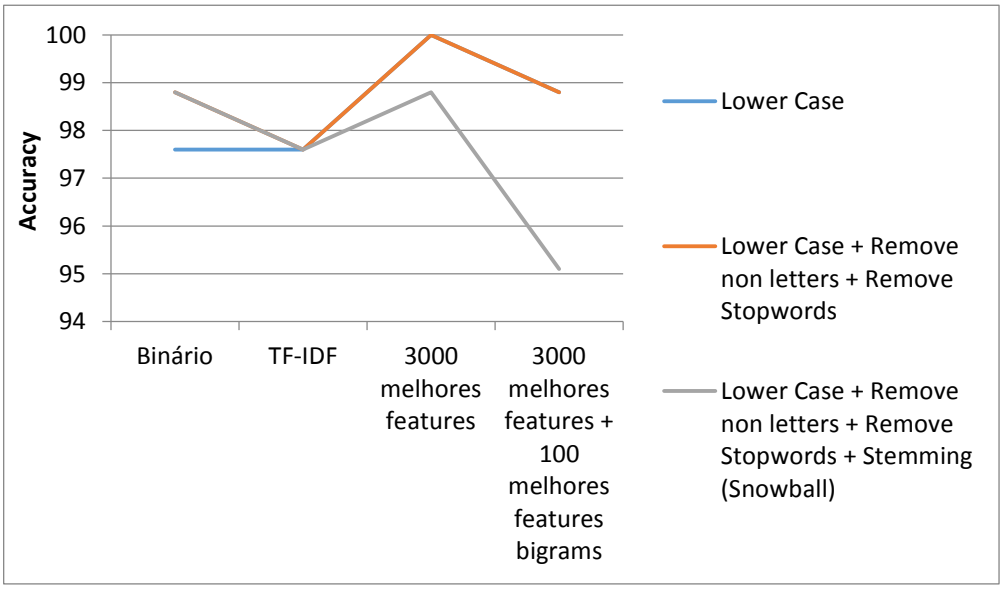


Figura 28 – Enzimas – Linear SVC

O melhor resultado com o classificador Linear SVC foi de 100% de *accuracy* com a escolha das 3000 melhores *features*, com a transformação para letras minúsculas e com remoção de *non letters* e *stopwords*.

De seguida é apresentado um gráfico com os resultados com o classificador Nu SVC do Scikit-Learn.

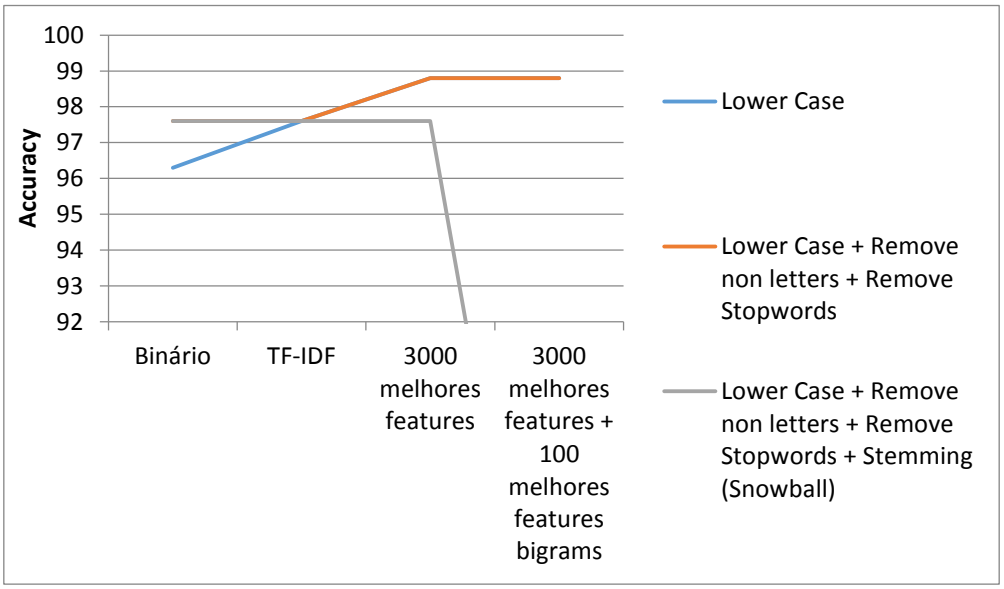


Figura 29 – Enzimas – Nu SVC

O melhor resultado com o classificador Nu SVC foi de 98.8% de *accuracy* com a escolha das 3000 melhores *features* ou das melhores *features* em conjunto com os melhores *bigrams*, e com transformação para letras minúsculas e remoção de *non letters* e *stopwords*.

De seguida é apresentado um gráfico com os resultados com o classificador *Logistic Regression* do Scikit-Learn.

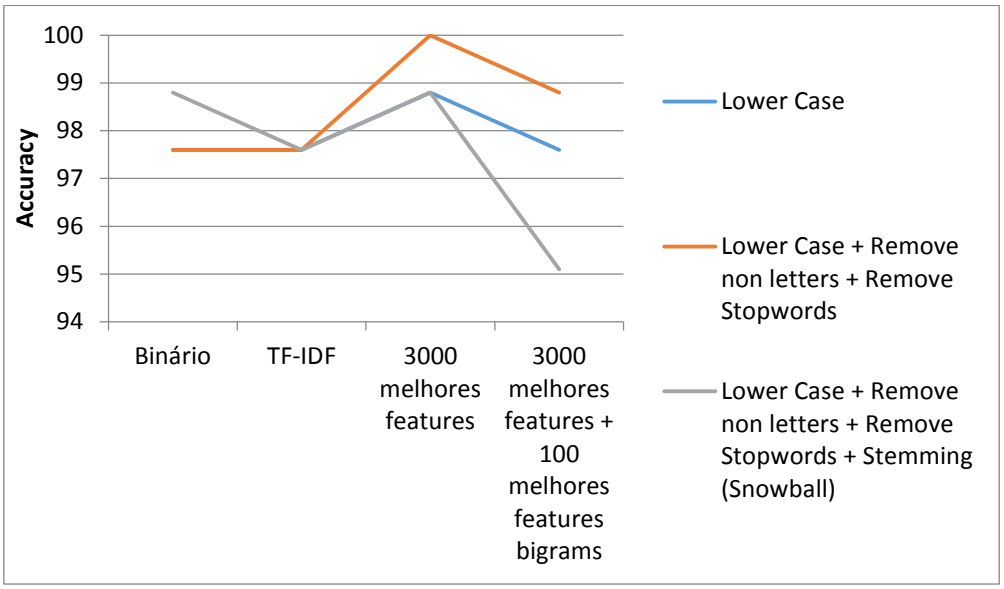


Figura 30 – Enzimas – Logistic Regression

O melhor resultado com o classificador *Logistic Regression* foi de 100% de *accuracy* com a escolha das 3000 melhores *features*, com a transformação para letras minúsculas e com remoção de *non letters* e *stopwords*.

Nos resultados obtidos com os diferentes classificadores é possível verificar que utilização das melhores *features* faz com que os resultados de *accuracy* estejam muito próximos dos 100% em todos classificadores e atingindo os 100% com o Linear SVC e *Logistic Regression*.

A principal conclusão, é que a redução da dimensão dos vetores com as melhores *features*, aumenta significativamente o desempenho e a qualidade dos modelos de classificação.

Os resultados obtidos mostram que a utilização do *stemming* juntamente com os restantes operadores de pré processamento diminui a qualidade dos resultados.

5 Aplicação Web

5.1 Ambiente de Desenvolvimento

O ambiente de desenvolvimento que serviu para a construção do projeto foi o seguinte:

- **Máquina de desenvolvimento**: Computador incluindo a máquina virtual para desenvolvimento da aplicação web.
- **Máquina virtual**: Máquina virtual incluido todas as ferramentas necessárias para a implementação da aplicação web, nomeadamente o servidor Glassfich, o sistema de gestão de base de dados (SGBD) MySQL, o IDE NetBeans e todas as ferramentas para desenvolvimento da plataforma J2EE, incluindo as bibliotecas necessárias.
    - Windows Server 2012 (64 bits)
    - Memória Ram – 8GB

5.2 Tecnologias Usadas

De seguida, são apresentadas as principais tecnologias usadas na implementação da aplicação web:

- **Linguagens de Programação**: Java, Python, SQL
- **Plataforma** J2EE
- **Páginas Web**: JSP, HTML
- Servlets
- **Servidor da aplicação**: Glassfish
- **Sistema de Gestão de Base de Dados**: MySQL
- **Mapeamento objeto relacional (ORM)**: JPA
- **HTTP Python Web Framework:** Bottle
- **Servidor HTTP:** CherryPyWSGIServer

5.3 Arquitetura da aplicação

A arquitetura representada neste Capítulo, apresenta detalhadamente os componentes da aplicação web e a sua interação com o código desenvolvido para a classificação de documentos em Python.

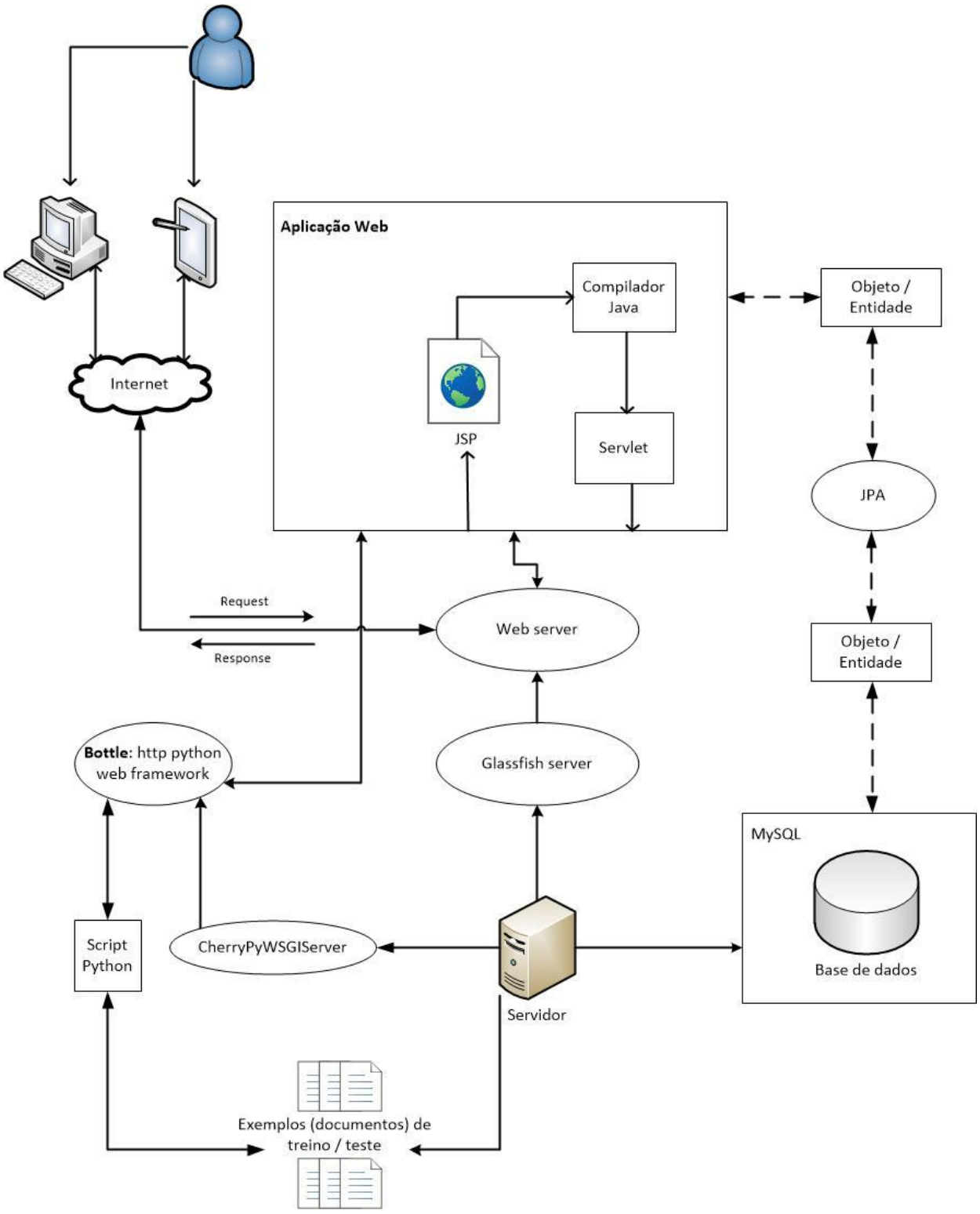


Figura 31 – Arquitetura detalhada

Os conjuntos de documentos para treino são guardados no servidor na diretoria C:/UploadDatasets. Os novos documentos que os utilizadores pretendam classificar

são temporariamente guardados na diretoria C:/UploadTestes e depois do processo de classificação terminar, são removidos da diretoria.

A aplicação web foi implementada com recurso a páginas JSP, uma tecnologia que permite gerar páginas dinâmicas, através de HTML, XML, JavaScript entre outras tecnologias, e usando a linguagem de programação Java. As páginas JSP são transformadas posteriormente em Servlets. As Servlets permitem a construção de aplicações do lado do servidor, e são executadas pelo mesmo como resposta a um pedido GET ou POST, sendo estas identificadas por um URL. As respostas das Servlets aos clientes são em linguagem HTML [42].

Para a utilização da base de dados relacional, foi usado o JPA, uma API que descreve uma interface comum para *frameworks* de persistência de dados. As tabelas da base de dados são representadas através de classes e os registros das tabelas são representados com instâncias das classes correspondentes. Com esta tecnologia, não é necessário implementar os comandos em linguagem SQL, pois disponibiliza uma interface de programação simples que realiza o trabalho de persistência [43].

O acesso da aplicação às funções do script em Python para todas as tarefas de classificação de documentos é realizado através da *framework* Bottle, e esta é que invoca diretamente os métodos (Capítulo 4.2 – Segunda versão do código) do script de classificação implementado. O Bottle é uma *micro-web framework* HTTP para Python, em que os métodos são invocados através de um URL HTTP, de forma muito semelhante a um *web service* [44]. Foram implementados dois métodos no Bottle, o método **treinar** e o método **classificar**.

O método **treinar** recebe todos os parâmetros necessários para ser treinado um modelo, invoca todas as funções necessárias do script de classificação implementado (pela ordem referida no Capítulo 4.2 – Segunda versão do código), e retorna a qualidade do modelo. Os parâmetros necessários ao invocar o método treinar são os seguintes:

-paramNomeDataset – Nome do *dataset* que é treinado.

-paramTipoTreinoTeste – Tipo de treino e teste a usar - 1 (70%/30%); 2 (*cross-validation*); 3 (cross-validation aleatório).

-paramOp1 – Indica se é para utilizar a remoção de *non letters* – True (Utilizar); False (Não Utilizar).

-paramOp2 – Indica se é para utilizar a remoção de *stopwords* – True (Utilizar); False (Não Utilizar).

-paramOp3 – Indica se é para utilizar a transformação dos *tokens* em letra minúscula – True (Utilizar); False (Não Utilizar).

-paramOp4 – Indica se é para utilizar o *stemming* – True (Utilizar); False (Não Utilizar).

-paramTipoClassificador – Tipo de classificador – 1 (NaiveBayesClassifier); 2 (MultinomialNB); 3 (BernoulliNB); 4 (NuSVC); 5 (LinearSVC); 6 (Logistic Regression).

-paramMelhoresPalavras – Seleção das melhores *features* - >0 (Utilizar as melhores X(valor do parâmetro) *features*); 0 (Utilizar todas as *features*); -1 (Utilizar todas as features com TF-IDF).

-paramMelhoresNGrams - Seleção das melhores *features bigrams* - >0 (Utilizar as melhores X(valor do parâmetro *features bigrams*); 0 (Não utilizar); -1 (Utilizar todas as *features* com TF-IDF).

-idClassificador – ID do classificador guardado na base de dados. Os modelos de classificação treinados são guardados em memória nesta *framework* através do ID, de maneira a que depois de ser treinado, o utilizador possa classificar novos documentos sem que o sistema tenha de voltar a treinar o modelo.

O método **classificar** permite classificar novos documentos, para tal, o método invoca todos os métodos implementados nos scripts concebidos para esta tarefa. Os parâmetros necessários ao invocar o método treinar são os seguintes:

- Todos os parâmetros usados no método treinar, em que se o ID estiver em memória apenas invoca o método de classificar documentos. Senão (caso o servidor vá abaixo, os classificadores são removidos da memória) volta a treinar, e de seguida classifica os documentos.

-pastaFicheiros – Nome da diretoria onde estão os documentos a classificar.

A *framework* Bottle está alojada num servidor HTTP chamado CherryPyWSGIServer, que possibilita a vários utilizadores em simultâneo utilizarem as operações de treino e de classificação através do Bottle [45].

5.4 Base de dados

De modo a armazenar os dados necessários do projeto foi implementada uma base de dados. Inicialmente, foi realizado o modelo conceptual da base de dados e de seguida o modelo físico. Após gerar o modelo físico foi obtido o ficheiro *crebas.sql* a partir do modelo físico com a ferramenta PowerDesigner. Este ficheiro contém todos os scripts necessários á criação da base de dados, que foram utilizados para gerar a base de dados no MySQL Server.

Neste tópico encontram-se representados o diagrama físico da base de dados na figura seguinte, assim como uma breve descrição de cada tabela. A informação detalhada de cada tabela assim que o modelo físico encontram-se incluídos em anexo (Anexo C – Base de Dados).

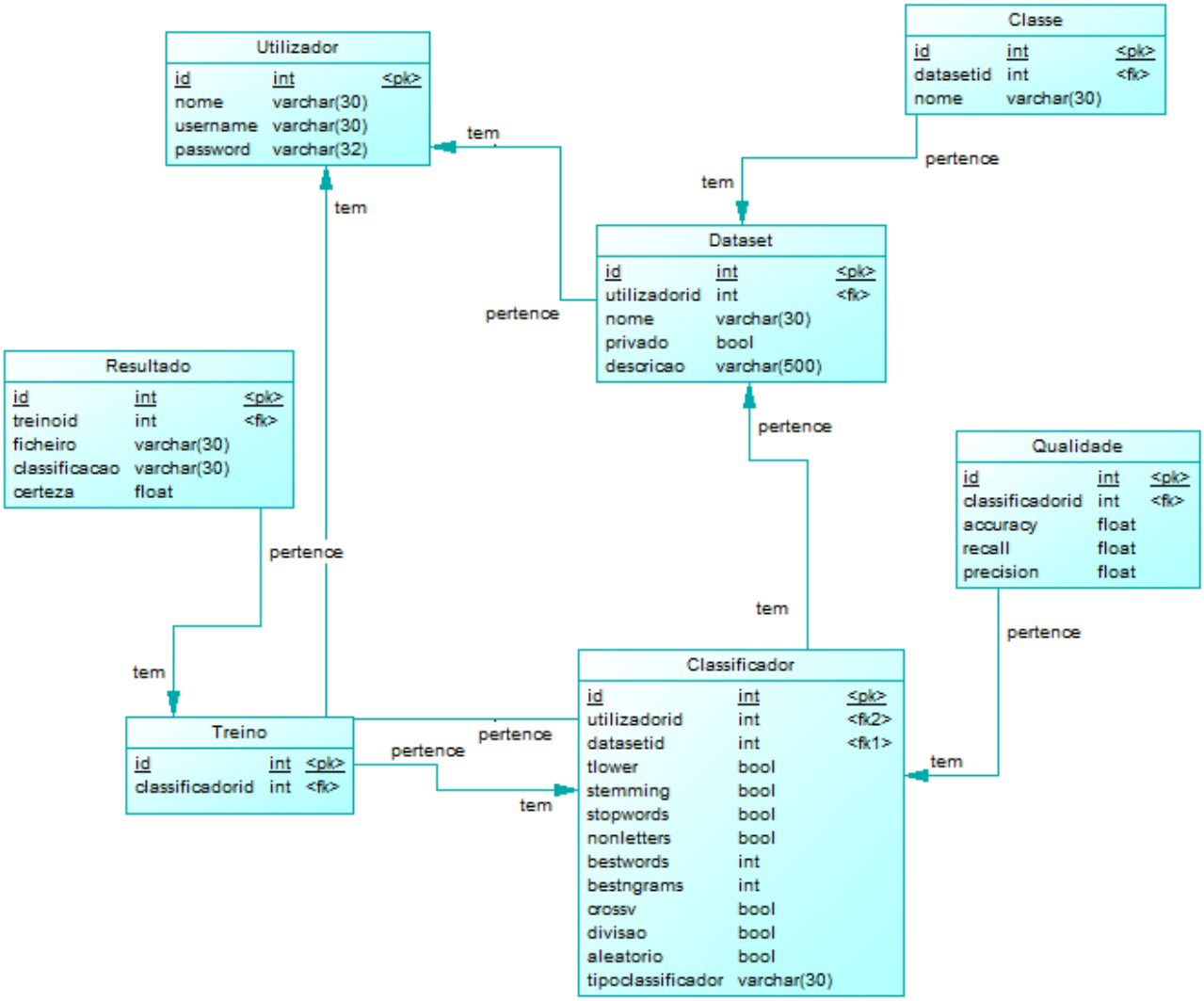


Figura 32 – Base de dados - Modelo físico

De seguida, é apresentado uma breve descrição das entidades:

- **Classe**

  Usada para armazenar as classes/categorias dos *datasets*. Como exemplo do *dataset* Movie Reviews, as classes seriam *neg* e *pos*. O *dataset* pode ter duas ou mais classes distintas. Cada classe pertence à entidade Dataset e tem uma chave estrangeira para a mesma.

- **Classificador**

Usada para armazenar os classificadores. Cada classificador é referente a um *dataset* e pertence a um utilizador tendo uma chave estrangeira para cada uma destas entidades. A entidade Classificador armazena as técnicas de pré processamento usadas (*lower case*; *stemming*; remoção de *non letters*; remoção de *stopwords*), o tipo de treino e de teste do classificador (percentagem 70% treino / 30% teste; *stratified cross-validation*; *stratified cross-validation* aleatório). Por fim esta entidade armazena ainda quais das técnicas é utilizada (Todos os *tokens* com TF-IDF; Todos os *tokens* binário; Palavras mais caracterizadoras e *bigrams* mais caracterizadores).

- **Dataset**

  Usada para armazenar os *datasets* para treino e teste. O *dataset* pertence a um utilizador tendo por isso uma chave estrangeira para a entidade Utilizador. Esta entidade armazena o nome do *dataset*, uma descrição (não obrigatória) e o estado do *dataset*: **privado** – acessível apenas para o utilizador que inseriu o mesmo; **público** – acessível para todos os utilizadores registados na plataforma.

- **Qualidade**

  Usada para armazenar a qualidade dos classificadores, contendo assim uma chave estrangeira para a entidade Classificador. Esta entidade guarda a *accuracy*, *recall* e *precision* do classificador, no caso de se usar o *cross-validation* ou do *dataset* ter mais de duas classes, os resultados guardados são a média de cada resultado.

- **Resultado**

  Usada para armazenar os resultados após a classificação de documentos com a classe ainda desconhecida. São registados o nome do ficheiro classificado, a classe na qual o ficheiro foi qualificado assim como a percentagem de certeza do ficheiro pertencer à classe em que foi classificado. Cada resultado pertence a um treino e tem uma chave estrangeira para a entidade Treino.

- **Treino**

  Usada para guardar os testes de um conjunto de documentos com a classe ainda desconhecida. Cada treino contem vários resultados (um por cada

ficheiro classificado), e pertence a um classificador, tendo uma chave estrangeira para a entidade Classificador.

- **Utilizador**

  Usada para armazenar as contas dos utilizadores. Nesta entidade são registados o *username*, o nome e a password encriptada com a função de dispersão criptográfica MD5 (*Message-Digest algorithm 5*). Um utilizador pode ter vários *datasets* e vários classificadores.

5.5 Testes

Neste projeto foram realizados testes de aceitação, com o objetivo de verificar o sistema em relação aos seus requisitos originais, e às necessidades do utilizador. Todos os testes passaram.

Os testes foram definidos num plano de testes. O "Anexo D – Testes" apresenta o plano de testes e os resultados dos testes.

5.6 Principais Interfaces

Os principais interfaces da aplicação web desenvolvidas são aqui apresentados. A aplicação apresenta um interface bastante simples de utilizar.

5.6.1 Menu principal

A aplicação contém um menu no topo sempre presente com várias opções que o utilizador pode realizar. No canto superior direito é possível realizar o login e logout da aplicação. A figura seguinte mostra o menu com as opções disponíveis.

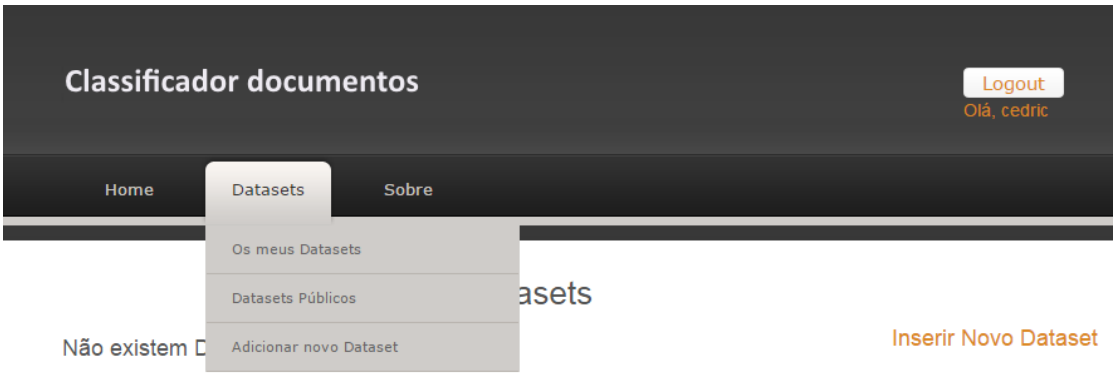


Figura 33 – Menu

A opção Home e Sobre estão disponíveis para qualquer utilizador, quer se tenha autenticado ou não. A opção Datasets e os seus submenus apenas estão disponíveis para utilizadores autenticados.

5.6.2 Adicionar Novo Dataset

Esta interface permite ao utilizador inserir um novo *dataset* privado ou público. Para inserir o *dataset*, é necessário escolher o ficheiro, a visibilidade do *dataset* (privado ou público), e opcionalmente adicionar uma descrição. A figura seguinte mostra o interface.

Figura 34 – Interface Adicionar novo dataset

5.6.3 Os Meus Datasets

Neste interface o utilizador tem acesso a todos os seus *datasets*. Pode realizar operações como ver os classificadores, adicionar um novo classificador ou remover o *dataset*.

O interface dos *datasets* públicos é semelhante a este com a única diferença a ser do utilizador não poder remover o *dataset*.

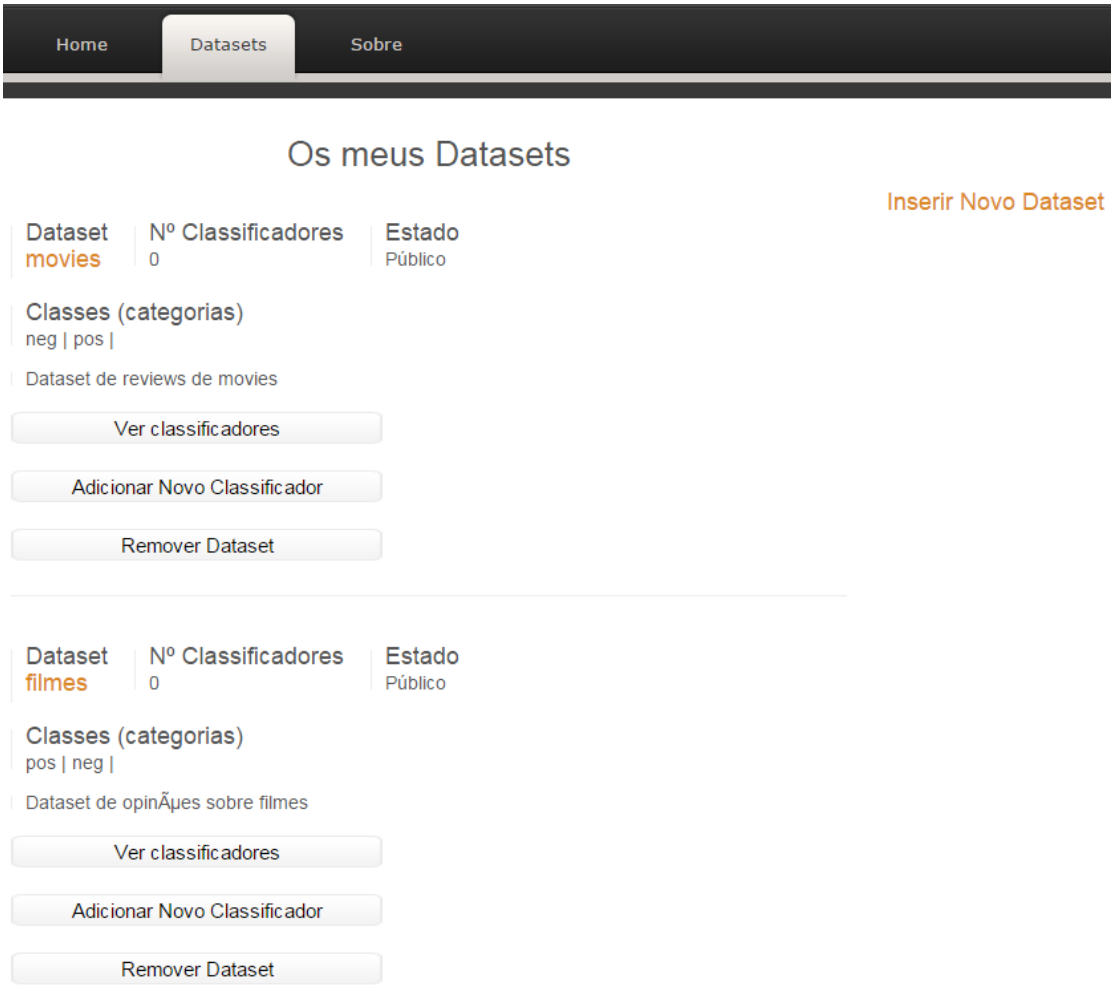


Figura 35 – Interface Os meus datasets

5.6.4 Adicionar Novo Classificador

Neste interface o utilizador tem a possibilidade de treinar um classificador a partir de um determinado *dataset*. O utilizador pode escolher o tipo de classificador, o tipo de treino, os operadores de pré-processamento, e as técnicas de *feature extraction*. A figura seguinte apresenta o interface.

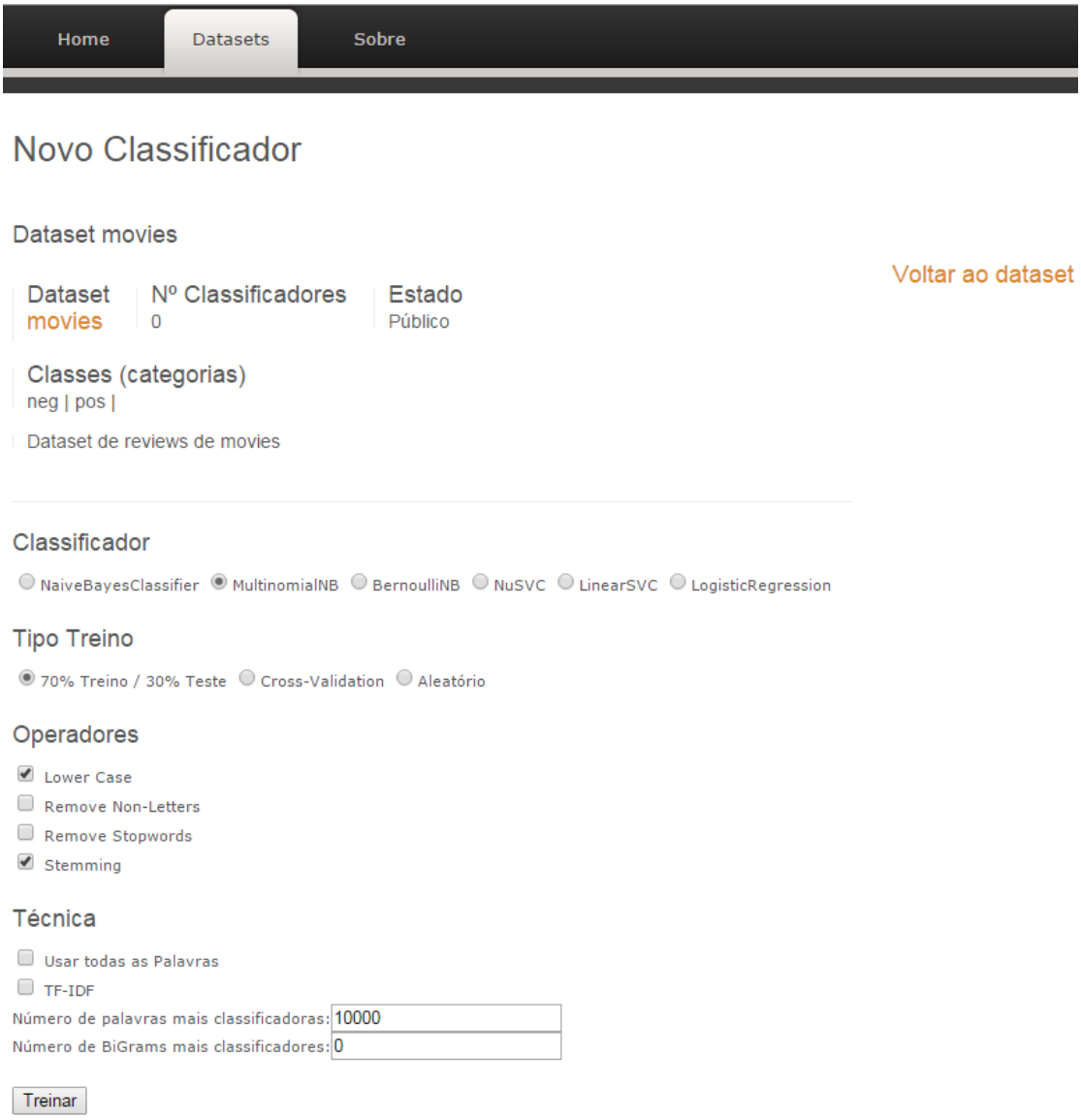


Figura 36 – Interface Adicionar novo classificador

5.6.5 Visualizar Classificadores

Interface onde se podem visualizar todos os modelos de classificação criados a partir de um determinado *dataset* e a qualidade do mesmo. O utilizador tem a possibilidade de aceder à página para classificar novos documentos, ver todos os resultados de classificação com um classificador, e remover o classificador. A figura seguinte apresenta o interface.

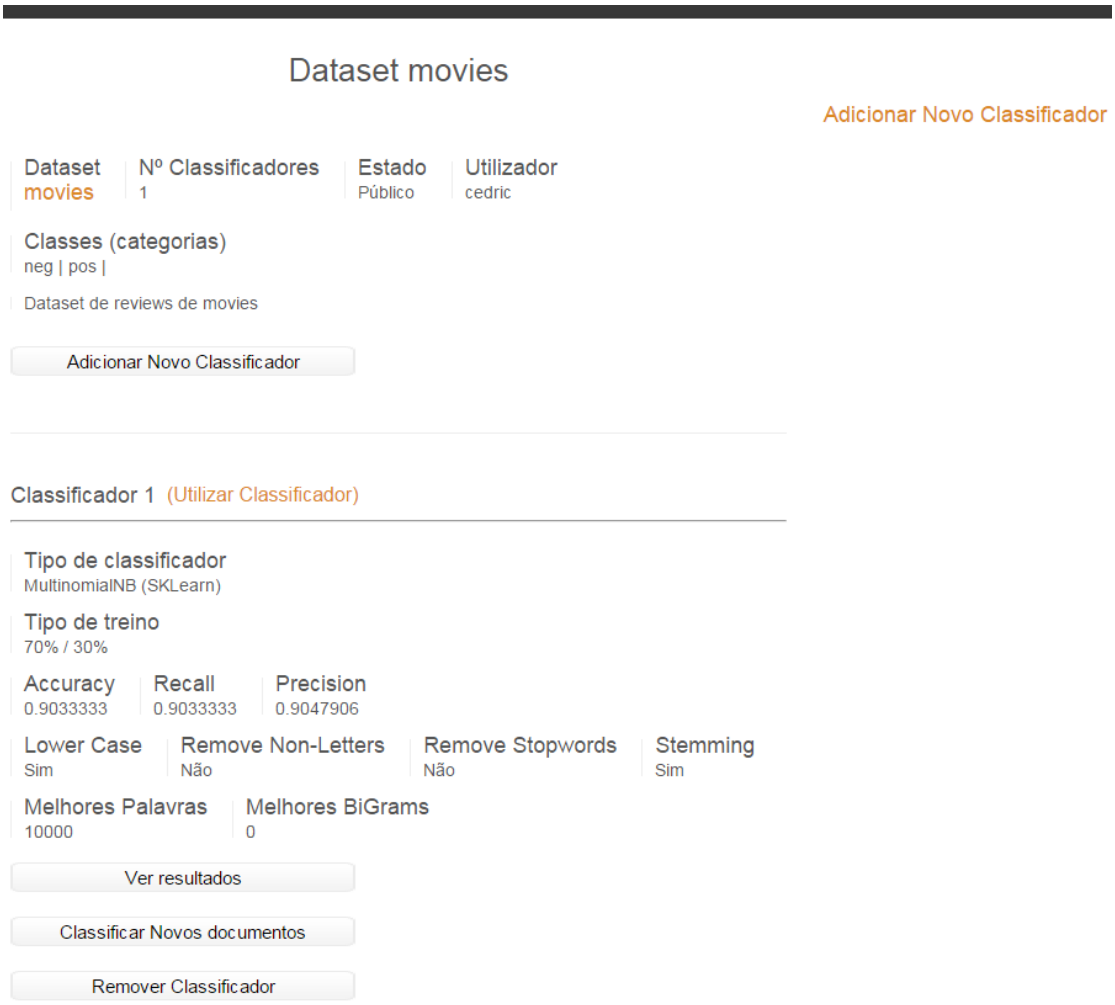


Figura 37 – Interface Visualizar classificadores

5.6.6 Classificador documentos

Interface para classificar documentos, a partir do classificador escolhido na página anterior. O utilizador faz o carregamento dos ficheiros através de um ficheiro em formato ".zip". A figura seguinte apresenta o interface.

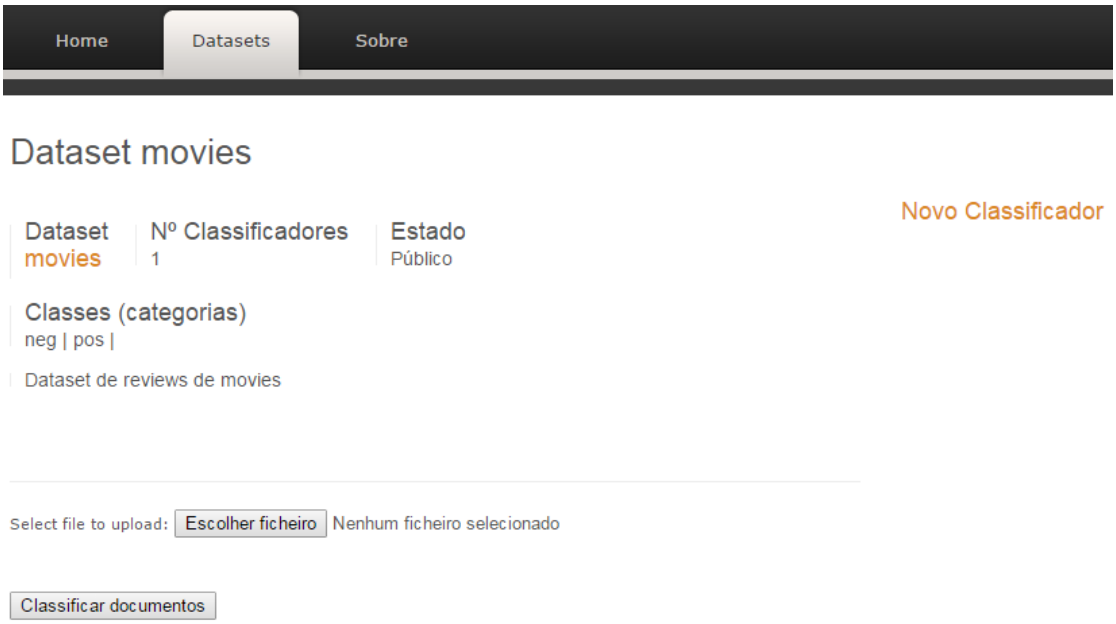


Figura 38 – Interface Classificar documentos

5.6.7 Visualizar Resultado de Classificação

Interface em que o resultado da classificação de documentos é apresentado ao utilizador. No resultado estão apresentados o *dataset*, o modelo de classificação, um gráfico com a percentagem de documentos categorizados em cada uma das classes, e o resultado da classificação de cada documento com a percentagem de certeza. O utilizador tem a possibilidade de remover o resultado, e de gerar um ficheiro em formato ".pdf" com o resultado. A figura seguinte apresenta o interface.

Resultado da classificação

Remover este resultado
Voltar aos resultados

Exportar resultado para PDF

Remover Resultado

Dataset

Dataset: movies
Estado: Público

Classes (categorias)
neg | pos |

Dataset de reviews de movies

Classificador (Utilizar Classificador)

Tipo de classificador
MultinomialNB (SKLearn)

Tipo de treino
70% / 30%

Accuracy	Recall	Precision
0.9033333	0.9033333	0.9047906

Lower Case	Remove Non-Letters	Remove Stopwords	Stemming
Sim	Não	Não	Sim

Melhores Palavras	Melhores BiGrams
10000	0

Resultados

Número de documentos classificados: 10
neg: 5
pos: 5

neg

cv998_15691.txt --> 99.99951% de certeza
cv997_5152.txt --> 99.999535% de certeza
cv996_12447.txt --> 99.93674% de certeza
cv999_14636.txt --> 99.99999% de certeza
cv995_23113.txt --> 100.0% de certeza

pos

cv997_5046.txt --> 100.0% de certeza
cv999_13106.txt --> 100.0% de certeza
cv998_14111.txt --> 100.0% de certeza
cv995_21821.txt --> 99.99997% de certeza
cv996_11592.txt --> 99.94663% de certeza

Figura 39 – Interface Visualizar resultado da classificação

6 Conclusões

O trabalho realizado teve como objetivo criar uma plataforma web para classificação de documentos baseado em algoritmos de aprendizagem supervisionada utilizando ferramentas de NLP e fazendo vários estudos de forma a avaliar os resultados dos algoritmos implementados. Os resultados obtidos revelaram uma muito boa performance, nomeadamente com a seleção das melhores *features*, reduzindo assim a dimensão dos vetores eliminando as *features* com pouco peso para classificação. Os resultados dos estudos com o *dataset* Movie Reviews selecionando as melhores *features* atingiram valores próximos de 93% de *accuracy*, superiores ao estudos referidos no Capítulo 2.3 em que o melhor resultado foi de 88.80% de *accuracy*. Os resultados com o *dataset* das Enzimas atingiram valores de 100% de *accuracy* com a utilização das melhores *features*.

Os resultados da classificação com o *dataset* Ohsumed de 23 classes apesar de baixarem um pouco em relação ao *dataset* Movie Reviews, o que era previsível pelo facto de ter 23 classes, foram muito bons com uma média de *accuracy* próxima dos 90%, muito acima dos valores registados nos estudos do Capítulo 2.3.

Os classificadores com melhores resultados nos diferentes casos de estudo foram o Linear SVC, o *Logistic Regression*, e Multinomial NB.

Em todos os casos de estudo realizados, a utilização de *stemming* não melhorou os resultados, sendo que na maioria dos resultados a qualidade baixou com este operador. Pelo contrário, a utilização dos operadores de remoção de *stopwords* e de *non letters* melhoraram os resultados ao invés de ser utilizado apenas a transformação para letra minúscula.

Com este projeto conclui-se ainda que a velocidade da execução de classificação com o NLTK e Scikit-Learn é de uma grande velocidade e eficiência, sendo que o código implemento consegue treinar um modelo e novos documentos em apenas alguns segundos.

6.1 Resultados do Projeto

Durante a realização do projeto, os resultados foram os scripts de classificação de documentos, a aplicação web assim como a respetiva Base de Dados, e a documentação produzida.

- **Scripts de Classificação**

  Os scripts implementados em Python com os algoritmos para classificação de documentos.

- **Aplicação Web**

  A aplicação web contém as funcionalidades previstas inicialmente. A principal função da aplicação é ser utilizada como interface do utilizador e utilizer os scripts desenvolvidos em Python de modo a utilizar as técnicas de classificação de documentos. Em conjunto com a aplicação foi também criada a base de dados com o objetivo de guarder os dados necessários.

- **Documentação produzida**

  A documentação produzida serve para a compreensão do projeto desenvolvido, o que também poderá ser útil para um futura remodelação da aplicação. Os documentos foram os seguintes:

  - **Documento "Anexo A – Casos de Uso" –** documento que contém o diagrama de casos de uso e a descrição dos casos de uso.
  - **Documento "Anexo B – Diagramas de Atividade" –** documento que contém os principais diagramas de atividade.
  - **Documento "Anexo C - Base de Dados"** – documento que contém todas as informações sobre a base de dados. Neste documento está presente o modelo concetual, o modelo físico e uma explicação detalhada das tabelas da base de dados utilizada no projeto.
  - **Documento "Anexo D - Testes" –** documento que contém o plano de testes e os testes realizados.
  - **Documento "Anexo E - Resultados de classificação de documentos com Python – 1ª versão (Movie Reviews)"** - documento com os

resultados obtidos na primeira versão do algoritmo. O caso de estudo foi realizado com o *dataset* Movie Reviews. (Entregue no CD)

- **Documento "Aenxo F - Resultados de classificação de documentos com Python – 2ª versão (Movie Reviews)** - documento com os resultados obtidos na segunda versão do algoritmo. O caso de estudo foi realizado com o *dataset* Movie Reviews. (Entregue no CD)
- **Documento "Aenxo G - Resultados de classificação de documentos com Python – 2ª versão (Ohsumed)"** - documento com os resultados obtidos na segunda versão do algoritmo. O caso de estudo foi realizado com o *dataset* Ohsumed. (Entregue no CD)
- **Documento "Aenxo H - Resultados de classificação de documentos com Python – 2ª versão (Enzimas)"** - documento com os resultados obtidos na segunda versão do algoritmo. O caso de estudo foi realizado com o *dataset* das Enzimas. (Entregue no CD)
- **Relatório de Projeto** – presente documento que tem por objetivo descrever todas as atividades realizadas ao longo do projeto.

6.2 Apreciação Final

Este projeto a nível do trabalho desenvolvido correu como planeado, tendo sido realizado o trabalho proposto e entregue todos os documentos e a aplicação com as funcionalidades pretendidas.

A nível pessoal foi sem dúvida um grande desafio ter realizado um projeto com um tema em que não tinha qualquer experiência e conhecimento. Desenvolver um projeto individualmente ao contrário de todos os trabalhos curriculares, realizados em grupo, foi um desafio aliciante e foi realizado com sucesso. Com este projeto sinto que adquiri novos conhecimentos, nomeadamente a nível da linguagem Python, e nas técnicas de processamento de texto. A utilização de J2EE na aplicação web permitiu-me aprofundar os meus conhecimentos nesta tecnologia. Os conhecimentos adquiridos vão sem dúvida ser úteis no meu futuro e estou muito grato por ter podido realizar esta dissertação.

6.3 Trabalho Futuro

A classificação de documentos através do processamento de linguagem natural é um tema atual em desenvolvimento, e que pode ser sempre melhorado e desenvolvido através de investigações na procura de novas técnicas e procedimentos. As propostas de trabalhos futuros a partir deste projeto são as seguintes:

- Possibilitar a utilização de *n-grams* com tamanho superior a dois, e utilização apenas dos mais importantes e categorizadores, e comparar os resultados com os obtidos através deste estudo.
- Ao serem usadas as melhores *features*, aquelas que melhor classificam as classes, possibilitar que o algoritmo escolha o número ótimo de melhores *features*. De recordar que neste momento o utilizador introduz manualmente os números que desejar.
- Permitir que ao utilizador, quando este cria um novo classificador, escolha o melhor classificador automaticamente. Isto é o algoritmo escolhe as melhores técnicas e o melhor classificador possível para obter a melhor performance possível para um determinado *dataset*.
- Utilizar novos classificadores e proceder a comparações com os classificadores já implementados.
- Procurar utilizar novas técnicas utilizando o NLTK e Scikit-Learn. As bibliotecas têm vindo a melhorar com sucessivas novas versões e novos métodos.
- Aprofundar as técnicas de NLP, realizando a análise semântica dos textos e para aplicar ao problema.
- Realizar automaticamente a classificação através de ficheiros obtidos a partir de API's disponibilizadas por páginas de literatura científica ou de outros tipos de artigos.
- Possibilitar a classificação de ficheiros de texto, com outros formatos, nomeadamente ficheiros de Word e ficheiros Pdf.

Referências

[1] *História da escrita*. (s.d.). Obtido em Novembro de 2015, de Wikipedia: https://pt.wikipedia.org/wiki/Hist%C3%B3ria_da_escrita

[2] *Understanding Information Overload*. (s.d.). Obtido em Novembro de 2015, de infogineering: http://www.infogineering.net/understanding-information-overload.htm

[3] Hearst, M. A. (1999). Unstangling text data mining. *ACL '99 Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pp. 3-10.

[4] Gupta, V., & Lehal, G. S. (2009). A Survey of Text Mining Techniques and Applications. *Journal of Emerging Technologies in Web Intelligence*, *vo.l 1, no. 1*, pp. 60-79.

[5] Patel, F. N., Soni, N. R. (2012). Text Mining: A Brief survey. *International Journal of Advanced Computer Research*, *vol. 2*, no. 4, pp. 76-85.

[6] Liddy, E. D. (2001). *Natural Language Processing.* Encyclopedia of Library and Information Science.

[7] Fischer, S. R. (2001). *History of Language.* Reaktion Books.

[8] *Natural language processing.* (s.d.). Obtido em Novembro de 2015, de Wikipedia: https://en.wikipedia.org/wiki/Natural_language_processing

[9] Feldman, S. (1999). NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval. *ONLINE –WESTON THEN WILTON*, *vol. 23*, no. 3, pp. 62-73.

[10] *Tipos de Data Mining*. (s.d.). Obtido em Novembro de 2015, de FEUP: http://paginas.fe.up.pt/~mgi99021/it/tipos.htm

[11] Gonçalves, C. A., Gonçalves, C. T., Camacho, R., & Oliveira, E. (2010). The Impact of Pre-processing on the Classification of MEDLINE Documents. *Proceedings of the 10th International Workshop on Pattern Recognition in Information Systems*, (pp. 53-61).

[12] Forman, G., Kirshenbaum, E. (2008). Extremely fast text feature extraction for classification and indexing. CIKM '08 *Proceedings of the 17th ACM conference on Information and knowledge management*, pp. 1221-1230.

[13] Joachims, T. (1997). A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categoriation. *ICML '97 Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 143-151.

[14] Gerard, S., Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, vol. 24, no. 5, pp. 513-523.

[15] Singh, S. R., Murthy, H. A., & Gonsalves, T. A. (2010). Feature Selection for Text Classification Based on Gini Coefficient of Inequality. *JMLR: Workshop and Conference Proceedings - Feature Selection in Data Mining*, *vol. 10*, pp. 76-85.

[16] Yang, J., Qu, Z., & Liu, Z. (2014). Improved Feature-Selection Method Considering theImbalance Problem in Categorization. *The Scientific World Journal*, *vol. 14*.

[17] Khosla, R., Howlett, R. J., & Jain, L. C. (2005). *Knowledge-Based Intelligent Information and Engineering Systems.* Springer.

[18] Kim, K. J. (2015). *Information Science and Applications.* Springer.

[19] *Cross Validation*. (s.d.). Obtido em Novembro de 2015, de Carnegie Mellon University - School of Computer Science: https://www.cs.cmu.edu/~schneide/tut5/node42.html

[20] *Cross-validation (statistics).* (s.d.). Obtido em Novembro de 2015, de Wikipedia: https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[21] *sklearn.cross_validation.StratifiedKFold*. (s.d.). Obtido em Novembro de 2015, de Scikit-Learn: http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedKFold.html

[22] Khan, A., Baharudin, B., Lee, L. H., & Khan, K. (2010). A Review of Machine Learning Algorithms for Text-Documents Classification. *JOURNAL OF ADVANCES IN INFORMATION TECHNOLOGY, vol. 1, no. 1*.

[23] Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing With Python.* O'Reilly Media.

[24] Jain, L. C., & Vemuri, V. R. (1998). *Industrial Applications of Neural Networks.* CRC Press.

[25] Gama, J., Carvalho, A., Faceli, K., Lorena, A., & Oliveira, M. (2015). *Extração de Conhecimento de Dados - Data Mining.* Edições Sílabo.

[26] Bors, A. G. (2001). Introduction of the Radial Basis Function (RBF) Networks. *Online Symposium for Electronics Engineers, vol. 1,* pp. 1-7.

[27] Keerthi, S. S., & Lin, C. J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation, vol. 15, no. 1,* pp. 1667-1689.

[28] *sklearn.svm.SVC*. (s.d.). Obtido em Novembro de 2015, de Scikit-Learn: http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[29] *sklearn.svm.LinearSVC*. (s.d.). Obtido em Novembro de 2015, de Scikit-Learn: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[30] *sklearn.svm.NuSVC*. (s.d.). Obtido em Novembro de 2015, de Scikit-Learn: http://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html#sklearn.svm.NuSVC

[31] *Data Mining Concepts*. (s.d.). Obtido em Novembro de 2015, de Oracle: https://docs.oracle.com/cd/E11882_01/datamine.112/e16808/classify.htm#DMCON035

[32] Powers, D. M. (2007). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation.

[33] *Python(x,y) - the scientific Python distribution.* (s.d.). Obtido em Novembro de 2015, de Python(x,y): https://python-xy.github.io/

[34] Mount, S., Shuttleworth, J., & Winder, R. (2008). *Python for Rookies: A First Course in Programming.* Cengage Learning.

[35] *Natural Language Toolkit*. (s.d.). Obtido em Novembro de 2015, de NLTK: http://www.nltk.org/

[36] *Choosing the right estimator*. (s.d.). Obtido em Novembro de 2015, de Scikit-Learn: http://scikit-learn.org/stable/tutorial/machine_learning_map/

[37] *Movie Review Data*. (s.d.). Obtido em Novembro de 2015, de Cornell University - Department of Computer Science: http://www.cs.cornell.edu/people/pabo/movie-review-data/

[38] *TEXT CATEGORIZATION CORPORA*. (s.d.). Obtido em Novembro de 2015, de Department of INFORMATION ENGINEERING AND COMPUTER SCIENCE: http://disi.unitn.it/moschitti/corpora.htm

[39] *MEROPS the Peptidase Database*. (s.d.). Obtido em Novembro de 2015, de Merops: https://merops.sanger.ac.uk/

[40] *PubMed*. (s.d.). Obtido em Novembro de 2015, de PubMed: http://www.ncbi.nlm.nih.gov/pubmed

[41] *Servlets and JSP Pages Best Practices*. (s.d.). Obtido em Novembro de 2015, de Oracle: http://www.oracle.com/technetwork/articles/java/servlets-jsp-140445.html

[42] *The Java Persistence API - A Simpler Programming Model for Entity Persistence*. (s.d.). Obtido em Novembro de 2015, de Oracle: http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html

[43] *Bottle: Python Web Framework*. (s.d.). Obtido em Novembro de 2015, de Bottle: http://bottlepy.org/docs/dev/index.html

[44] *CherryPy A Minimalist Python Web Framework*. (s.d.). Obtido em Novembro de 2015, de CherryPy: http://www.cherrypy.org/

[45] Chen, Y. (s.d). Classification of Moview Reviews using Character N-gram. *Projetct Report - CSE 6339 Intro to Computer Linguistics*.

[46] Narayanan, V., Arora, I., Bhatia, A. (2013). A Review of Machine Learning Algorithms for Text-Documents Classification. *Intelligent Data Engineering and Automated Learning IDEAL 2013 Lecture Notes in Computer Science, vol. 8206, pp. 194-201*.

[47] Liang, A. (2006). Rotten Tomatoes: Sentiment Classification in Movie Reviews. *CS 229.*

[48] Allotey, D. A. (2011). Sentiment Analysis And Classification Of Online Reviews using Categorical proportional Difference. *Deparment of Computer Science For The Degree of master of Science In Computer Science University of Regina.*

[49] Kalaivani, P., Shunmuganathan, K. L. (2013). Sentiment Classification of Moview Reviews by Supervised Machiine Learning Approaches. *Indian Journal of Computer Science & Enginnering, vol. 4,no. 4, pp. 285*.

[50] Inkpen, D., Kennedy, A. (2006). Sentiment Classification of Movie Reviews using Contextual Valence Shifters. *Computational Intelligence, vol. 22,no. 2, pp. 110-125*.

[51] Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *ECML '98 Proceedings of the 10th European Conference on Machine Learning, pp. 137-142.*

[52] Shankar, S., Karypis, G. (2000). A Feature Weight Adjustement Algorithm for Document Categorization. *KDD-2000 Workshio on Text Mining.*

Anexos

Anexo A – Casos de Uso

Anexo B – Diagramas de Atividade

Anexo C – Base de Dados

Anexo D – Testes

Os seguintes anexos foram entregues no CD junto com o formato digital do relatório:

Anexo E – Resultados de classificação de documentos com Python – 1ª versão (Movie Reviews)

Anexo F – Resultados de classificação de documentos com Python – 2ª versão

(Movie Reviews)

Anexo G – Resultados de classificação de documentos com Python – 2ª versão (Ohsumed)

Anexo H – Resultados de classificação de documentos com Python – 2ª versão (Enzimas)

Anexo A – Casos de Uso

A.1 - Diagrama de Casos de Uso

A seguinte figura representa o diagrama de casos de uso da aplicação.

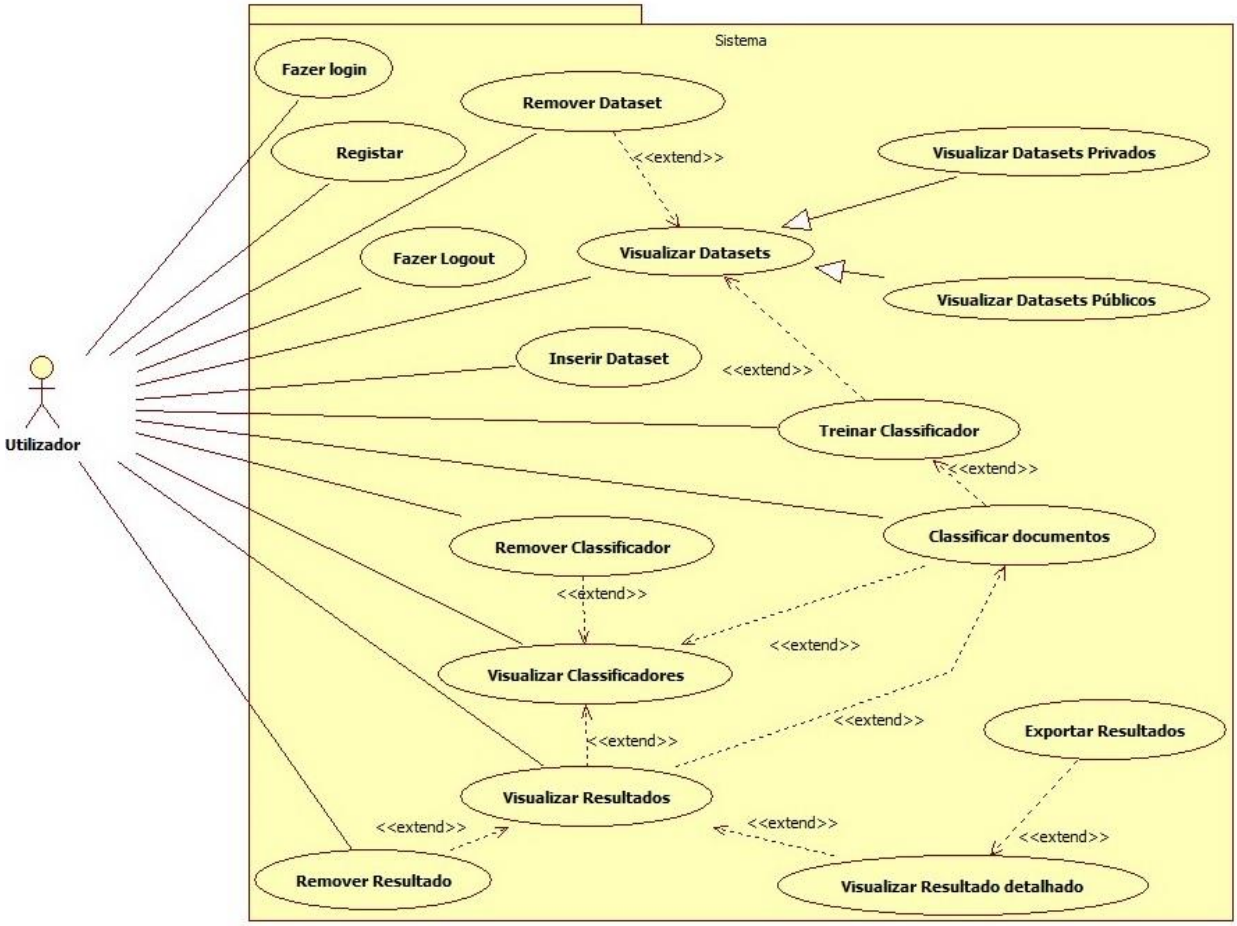


Figura 40 – Diagrama de Casos de Uso

A.2 - Escrita dos Casos de Uso

De seguida são descritos os casos de uso.

ID do caso de uso:	1
Caso de uso:	Registar
Actor:	Utilizador
Descrição:	O utilizador regista-se no sistema.
Pré-condições:	1. O utilizador não estar autenticado
Pós-condições:	1. Os dados do utilizador são guardados na base de dados.
Fluxo principal:	1. O utilizador acede à opção "Login" do menu principal. 2. O sistema apresenta o formulário de login. 3. O utilizador selecciona a opção "Novo registo". 4. O sistema apresenta o formulário de registo. 5. O utilizador preenche os dados e carrega no botão "Registar" para submeter. 6. O sistema verifica os dados. Caso o username já exista na base de dados apresenta uma mensagem de erro, e volta para o passo 4. 7. O sistema apresenta a página login.
Fluxo alternativo:	Nenhum

Tabela 17 – Caso de uso 1

ID do caso de uso:	2
Caso de uso:	Fazer login
Actor:	Utilizador
Descrição:	O utilizador autentica-se no sistema.
Pré-condições:	1. O utilizador não estar autenticado
Pós-condições:	1. O utilizador fica autenticado e apto a realizar as

	funcionalidades da aplicação.
Fluxo principal:	1. O utilizador acede à opção "Login" do menu principal. 2. O sistema apresenta o formulário de login. 3. O utilizador preenche o formulário. 4. O sistema verifica os dados. Caso estejam incorretos apresenta uma mensagem de erro e volta para o passo 2. 5. O sistema apresenta a página Home.
Fluxo alternativo:	Nenhum

Tabela 18 - Caso de uso 2

ID do caso de uso:	3
Caso de uso:	Fazer logout
Actor:	Utilizador
Descrição:	O utilizador faz logout.
Pré-condições:	1. O utilizador estar autenticado.
Pós-condições:	1. O utilizador sai da sua conta e fica sem permissões para aceder às principais funcionalidades da aplicação.
Fluxo principal:	1. O utilizador acede à opção "Logout" do menu principal. 2. O sistema apresenta a página Home.
Fluxo alternativo:	Nenhum

Tabela 19 - Caso de uso 3

ID do caso de uso:	4
Caso de uso:	Inserir Dataset
Actor:	Utilizador
Descrição:	O utilizador insere um dataset no sistema.
Pré-condições:	1. O utilizador estar autenticado

Pós-condições:	1. O dataset é armazenado no servidor, e os dados do mesmo são guardados na base de dados.
Fluxo principal:	1. O utilizador acede à opção "Adicionar Novo Dataset" do menu principal. 2. O sistema apresenta o formulário. 3. O utilizador selecciona o ficheiro do dataset, a visibilidade do mesmo, e uma descrição opcional. 4. O utilizador submete. 5. Se existir um dataset com o mesmo nome, volta para o passo 2. 6. O sistema apresenta a página de datasets.
Fluxo alternativo:	Nenhum

Tabela 20 – Caso de uso 4

ID do caso de uso:	5
Caso de uso:	Visualizar Datasets Privados
Actor:	Utilizador
Descrição:	O utilizador visualiza todos os seus datasets.
Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter inserido pelo menos um dataset.
Pós-condições:	Nenhuma.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador.
Fluxo alternativo:	Nenhum

Tabela 21 – Caso de uso 5

ID do caso de uso:	6
Caso de uso:	Visualizar Datasets Públicos
Actor:	Utilizador

Descrição:	O utilizador visualiza todos os datasets públicos.
Pré-condições:	1. O utilizador estar autenticado. 2. O sistema ter pelo menos um dataset público.
Pós-condições:	Nenhuma.
Fluxo principal:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos.
Fluxo alternativo:	Nenhum

Tabela 22 – Caso de uso 6

ID do caso de uso:	7
Caso de uso:	Remover Dataset
Actor:	Utilizador
Descrição:	O utilizador remove um dataset do sistema.
Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter inserido pelo menos um dataset.
Pós-condições:	1. O dataset é removido do servidor, e os dados do dataset e dos seus classificadores são removidos da base de dados.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador. 3. O utilizador selecciona a opção "Remover Dataset". 4. O sistema atualiza e a apresenta a lista dos datasets.
Fluxo alternativo:	Nenhum

Tabela 23 – Caso de uso 7

ID do caso de uso:	8
Caso de uso:	Treinar Classificador
Actor:	Utilizador

Descrição:	O utilizador treina um novo classificador
Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter inserido pelo menos um dataset ou existir pelo menos um dataset público.
Pós-condições:	1. É criado e treinado um classificador e é guardado na base de dados.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador. 3. O utilizador selecciona a opção "Adicionar Novo Classificador" no dataset que desejar. 4. O sistema apresenta um formulário. 5. O utilizador preenche o formulário com as opções desejadas. 6. O utilizador submete o formulário. 7. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão.
Fluxo alternativo:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos. 3. O utilizador selecciona a opção "Adicionar Novo Classificador" no dataset que desejar. 4. O sistema apresenta um formulário. 5. O utilizador preenche o formulário com as opções desejadas. 6. O utilizador submete o formulário. 7. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão.

Tabela 24 – Caso de uso 8

ID do caso de uso:	9
Caso de uso:	Visualizar Classificadores
Actor:	Utilizador
Descrição:	O utilizador visualiza todos os classificadores de um determinado dataset.
Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter treinado pelo menos classficador.

Pós-condições:	Nenhuma.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão.
Fluxo alternativo:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão.

Tabela 25 – Caso de uso 9

ID do caso de uso:	10
Caso de uso:	Remover Classificador
Actor:	Utilizador
Descrição:	O utilizador elimina um classificador criado por si.
Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter treinado pelo menos um classificador.
Pós-condições:	1. Os dados do classificador e os seus resultados são eliminados da base de dados.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Remover classificador" no classificador que desejar. 6. O sistema atualiza e apresenta a lista de classificadores.
Fluxo alternativo:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar.

	4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Remover classificador" no classificador que desejar. 6. O sistema atualiza e apresenta a lista de classificadores.

Tabela 26 – Caso de uso 10

ID do caso de uso:	11
Caso de uso:	Classificar documentos
Actor:	Utilizador
Descrição:	O utilizador classifica novos documentos com um classificador.
Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter treinado pelo menos um classificador.
Pós-condições:	1. Os resultados da classificação dos documentos são guardados na base de dados.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Classificar Novos documentos" no classificador que desejar. 6. O sistema apresenta um formulário para inserir os ficheiros. 7. O utilizador carrega os ficheiros e submete. 8. O sistema apresenta a página com os resultados detalhados.
Fluxo alternativo:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Classificar Novos documentos" no classificador que desejar. 6. O sistema atualiza e apresenta a lista de classificadores. 7. O utilizador carrega os ficheiros e submete. 8. O sistema apresenta a página com os resultados

	detalhados.

Tabela 27 – Caso de uso 11

ID do caso de uso:	12
Caso de uso:	Visualizar Resultados
Actor:	Utilizador
Descrição:	O utilizador visualiza todos os resultados de classificação de documentos com um determinado classificador.
Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter classificado documentos pelo menos uma vez.
Pós-condições:	Nenhuma.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Ver resultados" no classificador que desejar. 6. O sistema apresenta todos os resultados.
Fluxo alternativo:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Ver resultados" no classificador que desejar. 6. O sistema apresenta todos os resultados.

Tabela 28 – Caso de uso 12

ID do caso de uso:	13
Caso de uso:	Visualizar Resultado Detalhado

Actor:	Utilizador
Descrição:	O utilizador visualiza o resultado detalhado de uma classificação de documentos.
Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter classificado documentos pelo menos uma vez.
Pós-condições:	Nenhuma.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Ver resultados" no classificador que desejar. 6. O sistema apresenta todos os resultados. 7. O utilizador selecciona a opção "Ver Resultado Detalhado" no resultado que desejar. 8. O sistema apresenta o resultado detalhado.
Fluxo alternativo:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Ver resultados" no classificador que desejar. 6. O sistema apresenta todos os resultados. 7. O utilizador selecciona a opção "Ver Resultado Detalhado" no resultado que desejar. 8. O sistema apresenta o resultado detalhado.

Tabela 29 – Caso de uso 13

ID do caso de uso:	14
Caso de uso:	Exportar Resultados
Actor:	Utilizador
Descrição:	O utilizador exporta o resultado de uma classificação de documentos para um documento em formato ".pdf".

Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter classificado documentos pelo menos uma vez.
Pós-condições:	Nenhuma.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Ver resultados" no classificador que desejar. 6. O sistema apresenta todos os resultados. 7. O utilizador selecciona a opção "Ver Resultado Detalhado" no resultado que desejar. 8. O sistema apresenta o resultado detalhado. 9. O utilizador selecciona a opção "Exportar resultado para PDF" 10. O sistema apresenta o resultado detalhado em ".pdf".
Fluxo alternativo:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Ver resultados" no classificador que desejar. 6. O sistema apresenta todos os resultados. 7. O utilizador selecciona a opção "Ver Resultado Detalhado" no resultado que desejar. 8. O sistema apresenta o resultado detalhado. 9. O utilizador selecciona a opção "Exportar resultado para PDF" 10. O sistema apresenta o resultado detalhado em ".pdf".

Tabela 30 – Caso de uso 14

ID do caso de uso:	15
Caso de uso:	Remover Resultado
Actor:	Utilizador
Descrição:	O utilizador remove o resultado de uma classificação de documentos.
Pré-condições:	1. O utilizador estar autenticado. 2. O utilizador ter classificado documentos pelo menos uma vez.
Pós-condições:	Nenhuma.
Fluxo principal:	1. O utilizador acede à opção "Os meus Datasets" do menu principal. 2. O sistema apresenta todos os datasets do utilizador. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Ver resultados" no classificador que desejar. 6. O sistema apresenta todos os resultados. 7. O utilizador selecciona a opção "Remover Resultado". 8. O sistema atualiza e apresenta os resultados obtidos com o classificador em questão.
Fluxo alternativo:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Ver resultados" no classificador que desejar. 6. O sistema apresenta todos os resultados. 7. O utilizador selecciona a opção "Remover Resultado". 8. O sistema atualiza e apresenta os resultados obtidos com o classificador em questão.
Fluxo alternativo:	1. O utilizador acede à opção "Datasets Públicos" do menu principal. 2. O sistema apresenta todos os datasets públicos. 3. O utilizador selecciona a opção "Ver classificadores" no dataset que desejar. 4. O sistema apresenta todos os classificadores criados pelo utilizador com o dataset em questão. 5. O utilizador selecciona a opção "Ver resultados" no classificador que desejar.

	6. O sistema apresenta todos os resultados. 7. O utilizador selecciona a opção "Ver Resultado Detalhado" no resultado que desejar. 8. O sistema apresenta o resultado detalhado. 9. O utilizador selecciona a opção "Remover Resultado". 10. O sistema atualiza e apresenta os resultados obtidos com o classificador em questão.

Tabela 31 – Caso de uso 15

Anexo B – Diagramas de Atividade

Neste anexo, são apresentados os principais diagramas de atividade da aplicação.

B.1 – Inserir Dataset

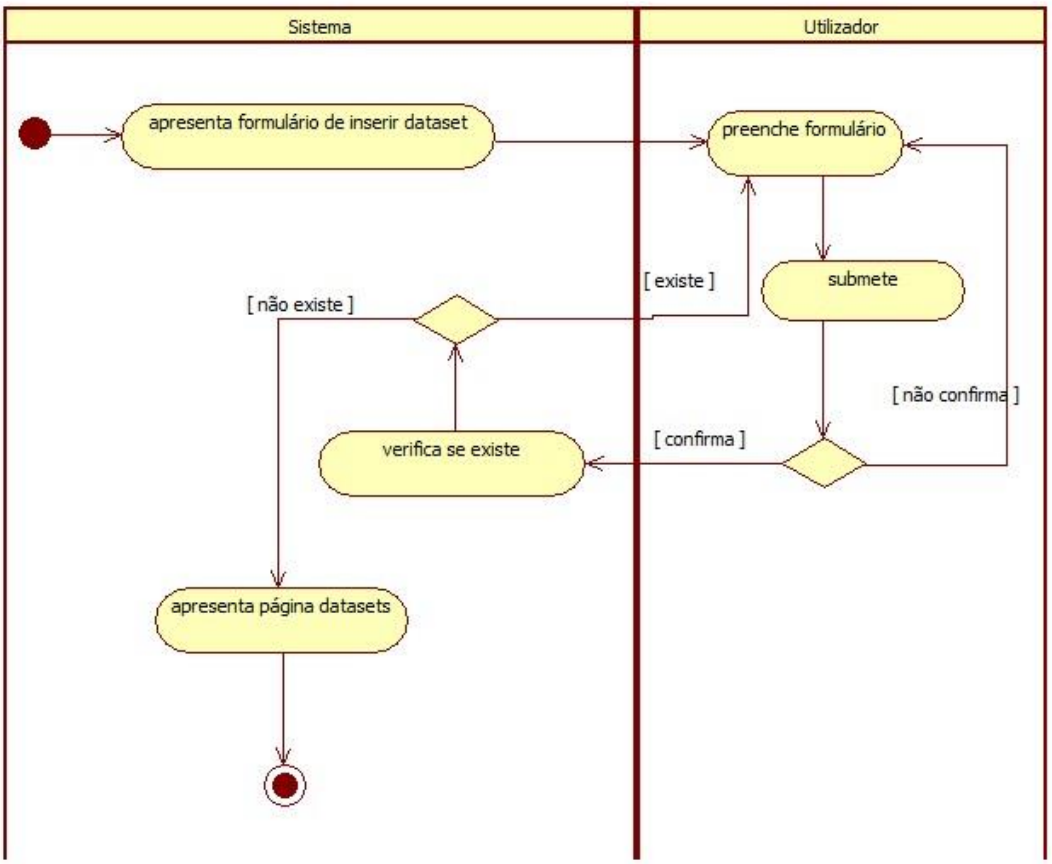


Figura 41 – Diagrama de Atividade – Inserir Dataset

B.2 – Treinar Classificador

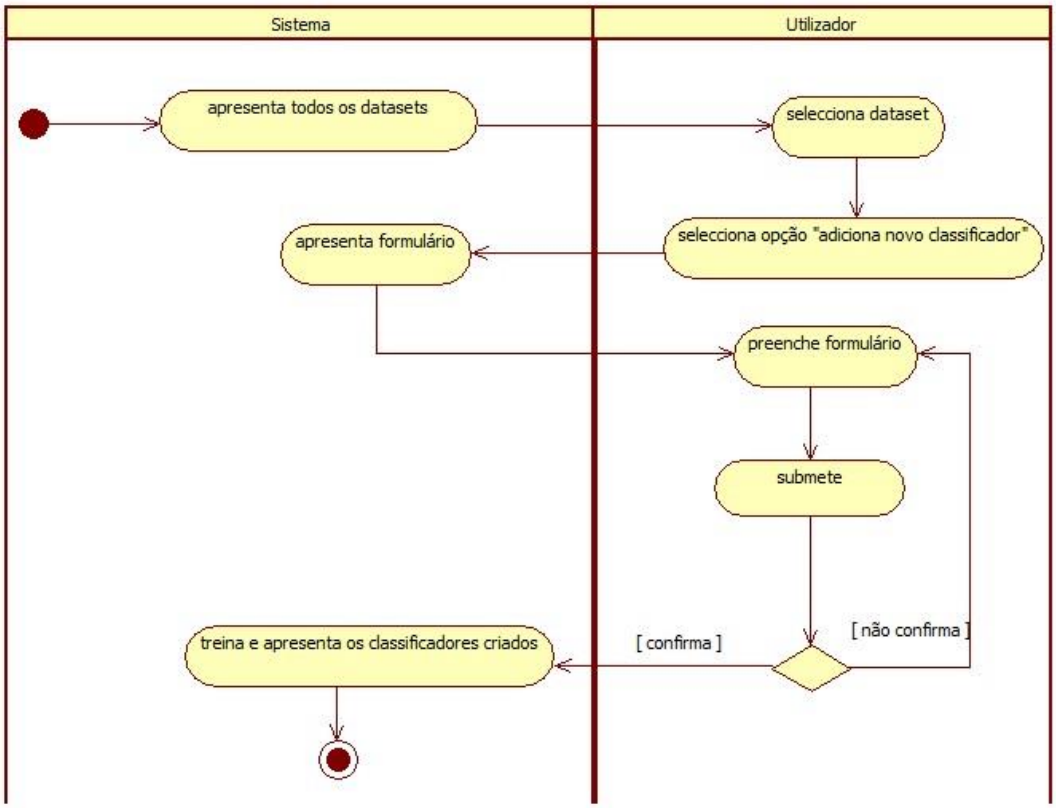


Figura 42 – Diagrama de Atividade – Treinar Classificador

B.3 – Classificar Documentos

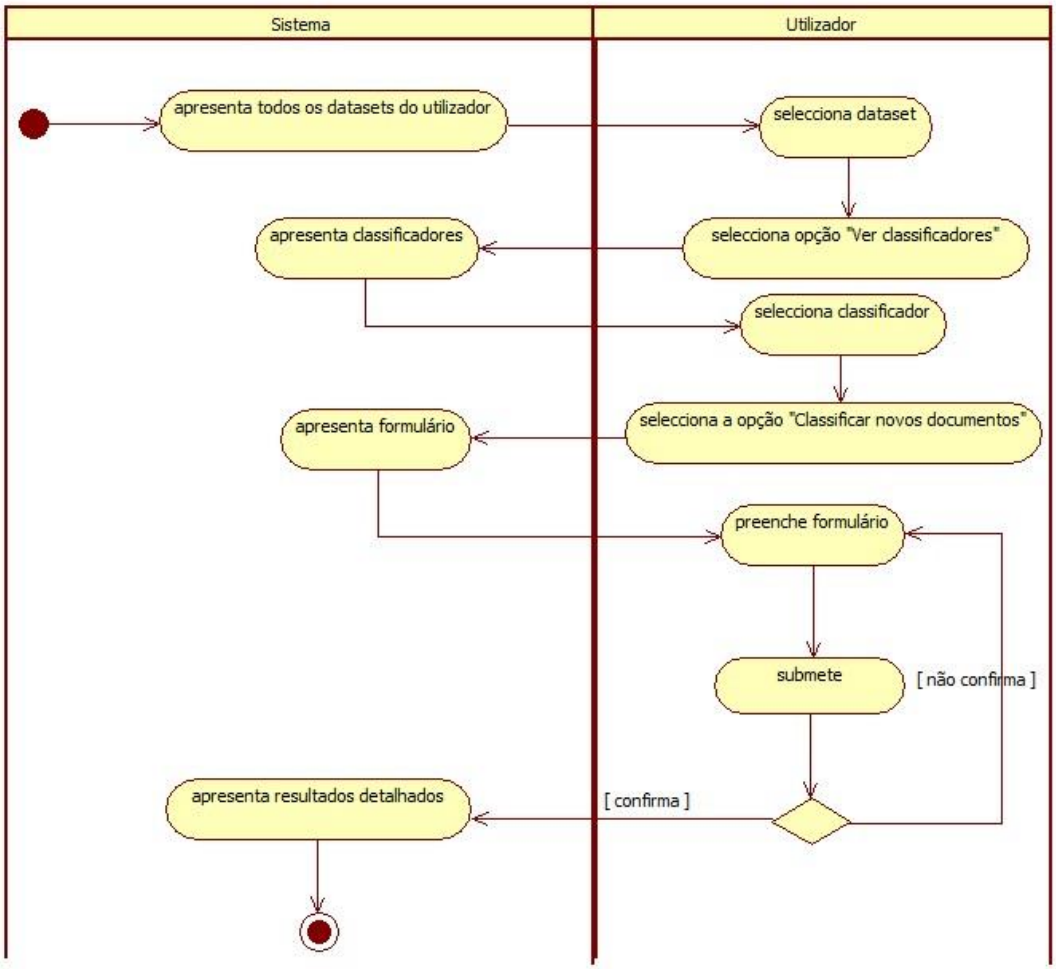


Figura 43 – Diagrama de Atividade – Classificar Documentos

B.4 – Visualizar Resultado Detalhado

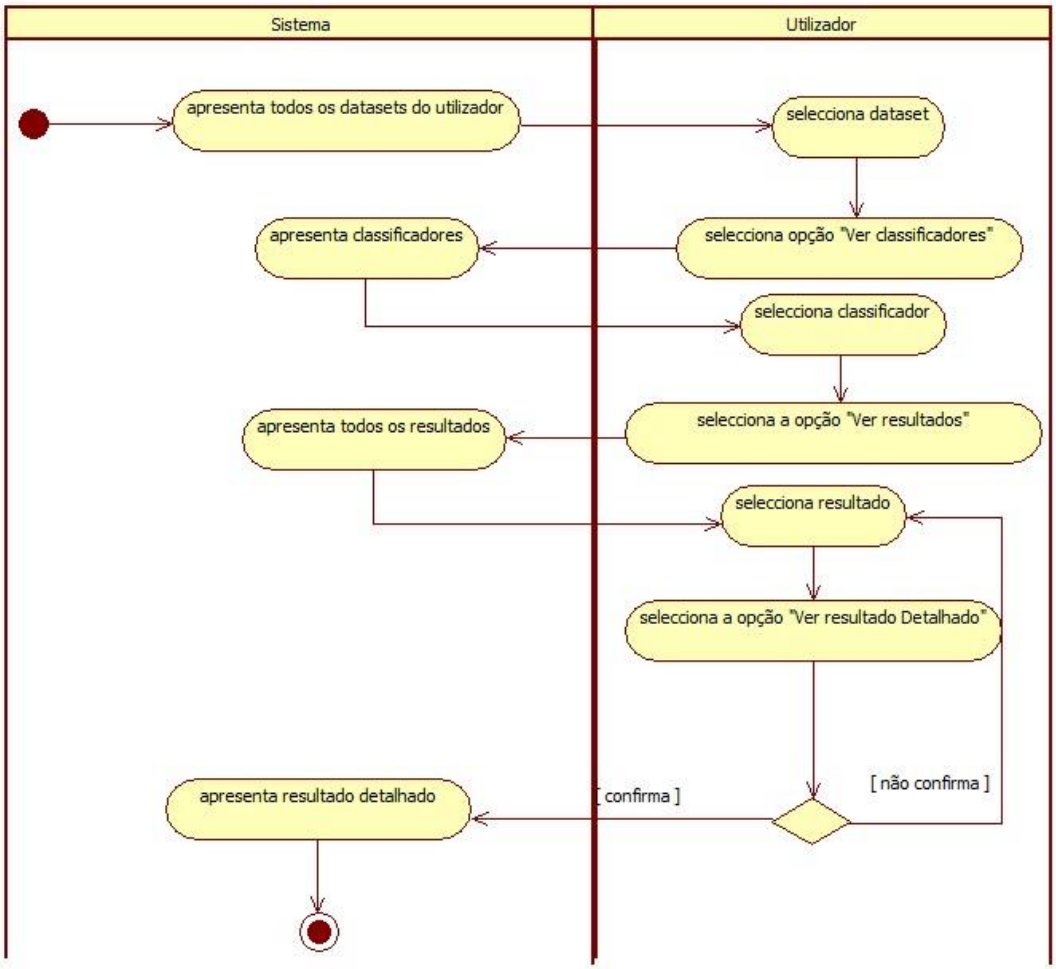


Figura 44 – Diagrama de Atividade – Visualizar Resultado Detalhado

Anexo C – Base de Dados

C.1 – Modelo Conceptual

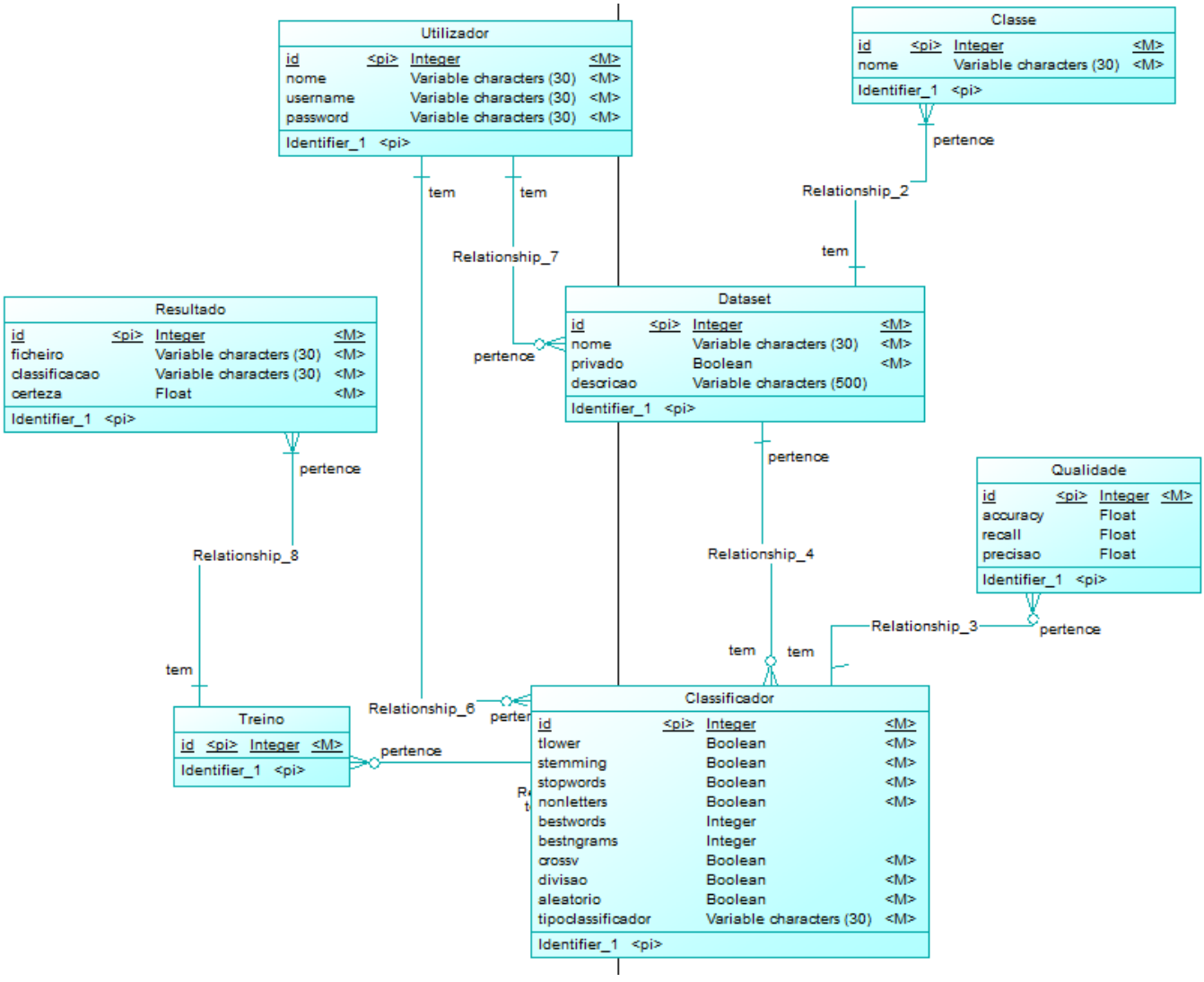


Figura 45 – Modelo Conceptual

C.2 – Modelo Físico

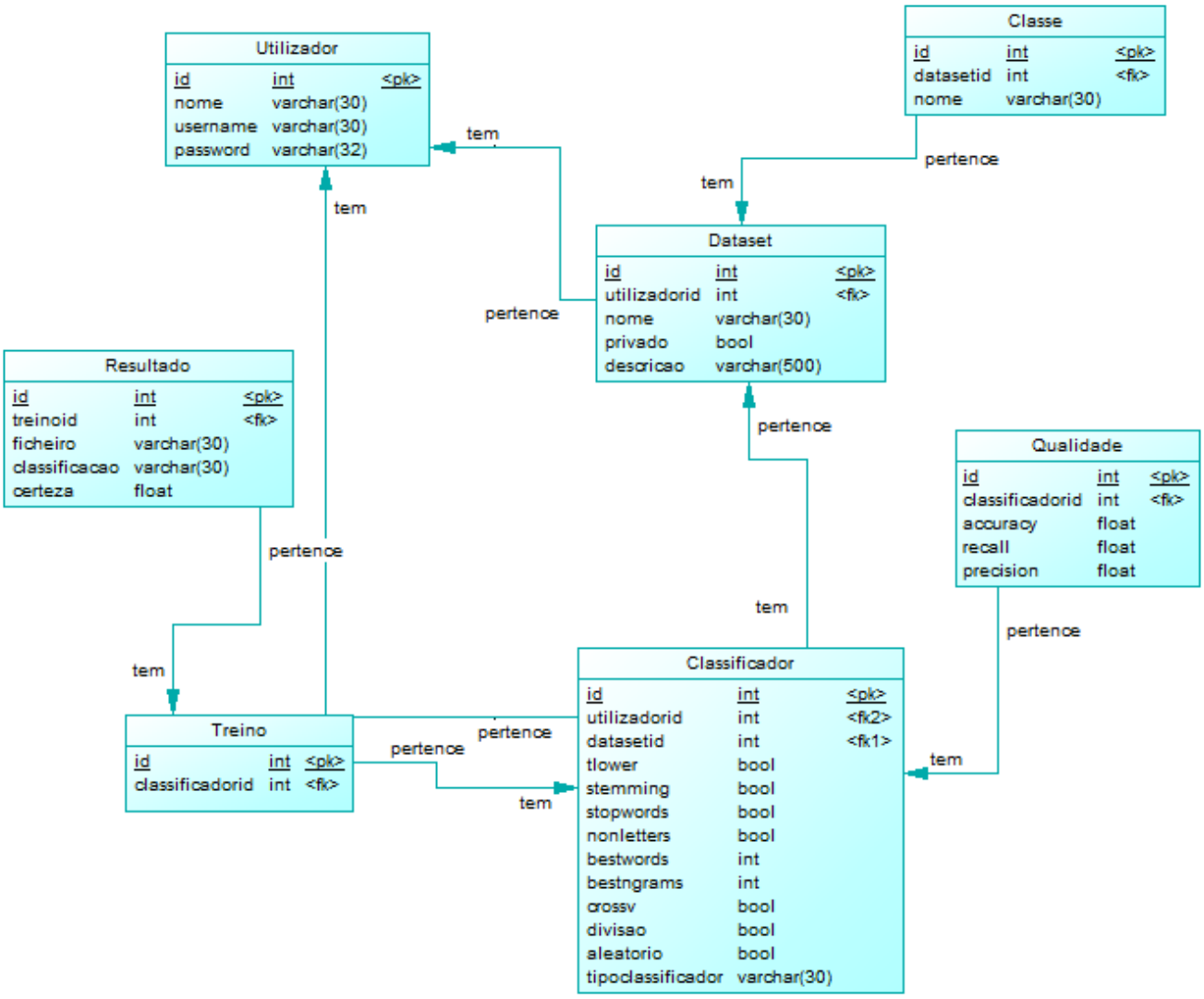


Figura 46 – Modelo Físico

C.2.1 – Tabelas

- Classe

Usada para armazenar as classes dos datasets.

Nome do atributo	Tipo de dados	Descrição
id	int	Chave primária
datasetid	int	Chave estrangeira
nome	Varchar(30)	Nome da classe

Tabela 32 – Tabela Classe

- Classificador

Usada para armazenar os classificadores.

Nome do atributo	Tipo de dados	Descrição
id	int	Chave primária
utilizadorid	int	Chave estrangeira
datasetid	int	Chave estrangeira
tlower	bool	Uso de lower case
stemming	bool	Uso de stemming
stopwords	bool	Uso de stopwords
nonletters	bool	Uso de nonletters
bestwords	int	Número de palavras mais definidoras
bestngrams	int	Número de bigrams mais definidores

crossv	bool	Uso de cross-validation
divisao	bool	Uso de percentagem 70% treino / 30% teste
aleatorio	bool	Uso de cross-validation aleatoriamente
tipoclassificador	varchar(30)	Tipo de classificador

Tabela 33 – Tabela Classificador

- Dataset

Usada para armazenar os datasets para treino e teste.

Nome do atributo	Tipo de dados	Descrição
id	int	Chave primária
utilizadorid	int	Chave estrangeira
nome	Varchar(30)	Nome do dataset
privado	bool	Estado do dataset (privado ou público)
descricao	Varchar(500)	Descrição do dataset

Tabela 34 – Tabela Dataset

- Qualidade

Usada para armazenar a qualidade dos classificadores.

Nome do atributo	Tipo de dados	Descrição
id	int	Chave primária

classificadorid	int	Chave estrangeira
accuracy	float	Accuracy do classificador
recall	float	Recall do classificador
precision	float	Precision do classificador

Tabela 35 – Tabela Qualidade

- Resultado

Usada para armazenar os resultados após a classificação de documentos com a classe ainda desconhecida.

Nome do atributo	Tipo de dados	Descrição
id	int	Chave primária
treinoid	int	Chave estrangeira
ficheiro	varchar(30)	Nome do ficheiro classificado
classificacao	varchar(30)	Classe na qual o ficheiro foi classificado
certeza	float	Percentagem de certeza do ficheiro pertencer à classe classificada

Tabela 36 – Tabela Resultado

- Treino

Usada para guardar os testes de um conjunto de documentos com a classe ainda desconhecida.

Nome do atributo	Tipo de dados	Descrição
id	int	Chave primária
classificadorid	int	Chave estrangeira

Tabela 37 – Tabela Treino

- Utilizador

Usada para armazenar as contas dos utilizadores.

Nome do atributo	Tipo de dados	Descrição
id	int	Chave primária
nome	varchar(30)	Nome do utilizador
username	varchar(30)	Username do utilizador
password	varchar(30)	Password do utilizador encriptada

Tabela 38 – Tabela Utilizador

Anexo D - Testes

Este anexo apresenta o plano de testes de aceitação, assim como os resultados dos mesmos.

D.1.1 – Estados base

Nome	**EB-1:** Efetua autenticação de um jogador.
Pré-condições	O utilizador estar registado.
Passos	1. O utilizador selecciona a opção "Login". 2. Introduz o nome de utilizador (cedric) e a password (123). 3. Submete o formulário.
Resultados esperados	1. O utilizador fica autenticado e a aplicação apresenta a página Home.

Tabela 39 – Estado Base 1

Nome	**EB-2:** Registo de um utilizador e inserção de um dataset público
Pré-condições	Nenhuna.
Passos	1. O utilizador selecciona a opção "Login". 2. O utilizador selecciona a opção "Novo registo". 3. Introduz o nome (joao), o username (joao), a password (123) e a confirmação da password (123). 4. O utilizador autentica-se introduzindo o username (joao) e password (123). 5. O utilizador selecciona a opção "Adicionar Novo Dataset". 6. Selecciona o ficheiro movies.zip para upload, o estado "Público" e a descrição (dataset de movies), e submete.
Resultados esperados	1. Na base de dados estão presentes os dados correspondentes ao utilizador registado, e ao dataset público inserido.

Tabela 40 – Estado Base 2

D.1.2 – Casos de teste

Nome	**CT-01:** Verificar se o sistema permite o registo de um utilizador.
Pré-condições	Nenhuma.
Passos	1. Selecciona a opção "Login". 2. Selecciona a opção "Novo registo". 3. Introduz o nome (carlos), o username (carlos), a password (123) e a confirmação da password (123).
Resultados esperados	1. O sistema apresenta a página de "Login" 2. O utilizador é acrescentado à base de dados.

Tabela 41 – CT-01

Nome	**CT-02:** Verificar se o sistema não permite registar dois utilizadores com o mesmo username.
Pré-condições	CT-01
Passos	1. Selecciona a opção "Login". 2. Selecciona a opção "Novo registo". 3. Introduz o nome (carlos), o username (carlos), a password (123) e a confirmação da password (123).
Resultados esperados	1. O sistema apresenta uma mensagem de erro a vermelho.

Tabela 42 – CT-02

Nome	**CT-03:** Verificar se o sistema apresenta a lista dos datasets públicos.
Pré-condições	EB-1,EB-2
Passos	1. Selecciona a opção "Ver Datasets Públicos".
Resultados esperados	1. O sistema apresenta o dataset público com o nome "movies".

Tabela 43 - CT-03

Nome	**CT-04:** Verificar se o sistema insere um dataset.
Pré-condições	EB-1
Passos	1. Selecciona a opção "Adicionar Novo Dataset". 2. Faz upload do ficheiro "filmes.zip", seleciona o estado "Privado" e insere a descrição "Dataset de filmes". 3. Submete.
Resultados esperados	1. O sistema apresenta a lista dos datasets, com o presente listado. 2. O dataset é inserido na base de dados. 3. O dataset é inserido na diretoria "C:/uploadDatasets"

Tabela 44 - CT-04

Nome	**CT-05:** Verificar se o sistema não permite a inserção de um dataset com o nome igual ao de outro dataset inserido.
Pré-condições	EB-1, CT-04
Passos	1. Selecciona a opção "Adicionar Novo Dataset". 2. Faz upload do ficheiro "filmes.zip", seleciona o estado "Privado" e insere a descrição "Dataset de filmes". 3. Submete.
Resultados esperados	1. O sistema apresenta uma mensagem de erro a vermelho. 2. Não é inserido na base de dados.

Tabela 45 – CT-05

Nome	**CT-06:** Verificar se o sistema adiciona um novo classificador.
Pré-condições	EB-1, CT-04
Passos	1. Selecciona a opção "Os Meus Datasets". 2. Selecciona a opção "Adicionar Novo Classificador" do dataset "filmes". 3. Deixar as opções do configurador de um novo classificador como "default" e clicar no botão "Treinar".
Resultados esperados	1. O sistema apresenta a lista de classificadores do dataset, com o classificador criado presente. 2. O classificador e a qualidade do mesmo são inseridos na base de dados.

Tabela 46 – CT-06

Nome	**CT-07:** Verificar se o sistema classifica documentos com um

	classificador.
Pré-condições	EB-1, CT-06
Passos	1. Selecciona a opção "Os Meus Datasets". 2. Selecciona a opção "Ver classificadores" do dataset "filmes". 3. Selecciona a opção "Classificar Novos documentos" de um classificador. 4. Carrega um ficheiro .zip com dois ficheiros ".txt" de opinião sobre filmes e seleciona o botão "Classificar documentos". 5. Seleccionar a opção "Exportar resultado para PDF".
Resultados esperados	1. O sistema apresenta o resultado da classificação dos documentos detalhada em PDF. 2. Os resultados da classificação dos documentos são inseridos na base de dados.

Tabela 47 – CT-07

Nome	**CT-08:** Verificar se o sistema remove um resultado de uma classificação de documentos.
Pré-condições	EB-1, CT-07
Passos	1. Selecciona a opção "Os Meus Datasets". 2. Selecciona a opção "Ver classificadores" do dataset "filmes". 3. Selecciona a opção "Ver resultados" do classificador utilizado no teste CT-07. 4. Clicar no botão "Remover Resultado" do resultado do teste CT-07.
Resultados esperados	1. O sistema apresenta a lista de resultados atualizada, sem o resultado eliminado. 2. O resultado é eliminado da base de dados.

Tabela 48 – CT-08

Nome	**CT-09:** Verificar se o sistema remove um classificador.
Pré-condições	EB-1, CT-06
Passos	1. Selecciona a opção "Os Meus Datasets". 2. Selecciona a opção "Ver classificadores" do dataset "filmes". 3. Selecciona a opção "Remover Classificador" do classificador criado no teste CT-06. 4. Seleccionar o botão Ok na mensagem de confirmação de

	remoção do classificador.
Resultados esperados	1. O sistema apresenta a lista de classificadores atualizada, sem o classificador eliminado. 2. O classificador é eliminado da base de dados.

Tabela 49 – CT-09

Nome	**CT-10:** Verificar se o sistema remove um dataset.
Pré-condições	EB-1, CT-04
Passos	1. Selecciona a opção "Os Meus Datasets". 2. Selecciona a opção "Remover Dataset" do dataset "filmes". 3. Seleccionar o botão Ok na mensagem de confirmação de remoção do dataset.
Resultados esperados	1. O sistema apresenta a lista de datasets atualizada, sem o dataset eliminado. 2. O dataset é eliminado da base de dados. 3. O dataset é eliminado da diretoria "C:/uploadDatasets".

Tabela 50 – CT-10

Nome	**CT-11:** Verificar se o sistema permite o logout de um utilizador.
Pré-condições	EB-1.
Passos	1. Selecciona a opção "Logout".
Resultados esperados	1. O sistema remove do menu a opção "Datasets". 2. O sistema apresenta a opção Login ao invés de Logout.

Tabela 51 – CT-11

D.2.1 – Resultados dos testes

Teste	Passou/Falhou
CT-01	Passou
CT-02	Passou
CT-03	Passou
CT-04	Passou
CT-05	Passou
CT-06	Passou
CT-07	Passou
CT-08	Passou
CT-09	Passou
CT-10	Passou
CT-11	Passou

Tabela 52 – Resultados dos testes

Classificação de Documentos com Processamento de Linguagem Natural

Anexo E – Resultados da classificação de documentos com Python - 1ª versão (Movie Reviews)

Índice

Índice de Tabelas

Introdução

Este documento tem como objetivo divulgar os resultados da classificação de documentos com a primeira versão do código em Python. O *dataset* para treino foi o seguinte: NLTK Movie Reviews Corpus. Este conjunto de documentos é relativo à crítica de filmes e é composto por 2000 documentos divididos em duas classes: crítica positiva (1000 documentos) e crítica negativa (1000 documentos). Todos os testes foram realizados com as 2000 palavras mais frequentes do corpus.

1 Resultados

1.1 Cross-Validation (4 subsets)

1.1.1 Binário

1.1.1.1 Sem operadores

Operadores usados					Cálculo Frequência		Classificação			Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X						X	Decision Tree	Cross-Validation		Pos – 0.733 Neg – 0.587	0.661	Pos – 0.640 Neg – 0.687
X						X	Maximum Entropy (algoritmo iis)	Cross-Validation		Erro de memória		
X						X	Maximum Entropy (algoritmo gis)	Cross-Validation		Pos – 0.951 Neg – 0.439	0.688	Pos – 0.642 Neg – 0.925
X						X	Logistic Regression	Cross-Validation		Pos – 0.858 Neg – 0.868	0.864	Pos – 0.866 Neg – 0.860
X						X	NB Bernoulli	Cross-Validation		Pos – 0.683	0.792	Pos – 0.873

									Neg – 0.901		Neg – 0.740
X						X	NB Multinomial	Cross-Validation	Pos – 0.803 Neg – 0.859	0.831	Pos – 0.85 Neg – 0.813
X						X	Linear SVC (SVM)	Cross-Validation	Pos – 0.853 Neg – 0.859	0.857	Pos – 0.858 Neg – 0.855
X						X	Nu SVC (SVM)	Cross-Validation	Pos – 0.852 Neg – 0.873	0.863	Pos – 0.870 Neg – 0.856
X						X	SVC (SVM)	Cross-Validation	Pos – 0.5 Neg – 0.5	0.479	Pos - Neg -

Tabela 1 - Binário sem operadores - Cross-Validation

1.1.1.2 Com operadores

Operadores usados					Cálculo Frequência		Classificação			Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X	X	X	X			X	Decision Tree	Cross Validation		Pos – 0.719 Neg – 0.601	0.659	Pos – 0.643 Neg – 0.682
X	X	X	X			X	Maximum Entropy (algoritmo iis)	Cross-Validation		Erro de memória		
X	X	X	X			X	Maximum Entropy (algoritmo gis)	Cross-Validation		Pos – 0.963 Neg – 0.408	0.680	Pos – 0.629 Neg – 0.935
X	X	X	X			X	Logistic Regression	Cross-Validation		Pos – 0.855 Neg – 0.863	0.86	Pos – 0.862 Neg – 0.857
X	X	X	X			X	NB Bernoulli	Cross-Validation		Pos – 0.699 Neg – 0.878	0.789	Pos – 0.851 Neg – 0.745
X	X	X	X			X	NB Multinomial	Cross-Validation		Pos – 0.817	0.826	Pos – 0.833

									Neg – 0.836		Neg – 0.819
X	X	X	X			X	Linear SVC (SVM)	Cross-Validation	Pos – 0.837 Neg – 0.861	0.850	Pos – 0.857 Neg – 0.842
X	X	X	X			X	Nu SVC (SVM)	Cross-Validation	Pos – 0.844 Neg – 0.880	0.863	Pos – 0.876 Neg – 0.850
X	X	X	X			X	SVC (SVM)	Cross-Validation	Pos – 0.5 Neg – 0.5	0.479	Pos - Neg -

Tabela 2 – Binário com operadores – Cross-Validation

1.1.1.3 Com operadores e Prune percentual (<3%)

Operadores usados					**Cálculo Frequência**		**Classificação**			**Valores obtidos**		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X	X	X	X	3%		X	Decision Tree	Cross-Validation		Pos – 0.737	0.673	Pos – 0.654

									Neg – 0.610		Neg – 0.698
X	X	X	X	3%		X	Maximum Entropy (algoritmo iis)	Cross-Validation	Pos – 0.591 Neg – 0.67	0.631	Pos – 0.645 Neg – 0.622
X	X	X	X	3%		X	Maximum Entropy (algoritmo gis)	Cross-Validation	Pos – 0.975 Neg – 0.328	0.646	Pos – 0.603 Neg – 0.940
X	X	X	X	3%		X	Logistic Regression	Cross-Validation	Pos – 0.827 Neg – 0.823	0.826	Pos – 0.825 Neg – 0.826
X	X	X	X	3%		X	NB Bernoulli	Cross-Validation	Pos – 0.748 Neg – 0.837	0.793	Pos – 0.821 Neg – 0.769
X	X	X	X	3%		X	NB Multinomial	Cross-Validation	Pos – 0.811 Neg – 0.838	0.825	Pos – 0.833 Neg – 0.817
X	X	X	X	3%		X	Linear SVC (SVM)	Cross-Validation	Pos – 0.809 Neg – 0.811	0.811	Pos – 0.810 Neg – 0.81
X	X	X	X	3%		X	Nu SVC (SVM)	Cross-Validation	Pos – 0.826 Neg – 0.845	0.837	Pos – 0.843 Neg – 0.831
X	X	X	X	3%		X	SVC (SVM)	Cross-Validation	Pos – 0.762 Neg – 0.885	0.822	Pos – 0.868 Neg – 0.788

Tabela 3 – Binário com operadores e Prune – Cross-Validation

1.1.2 TF-IDF

1.1.2.1 Sem operadores

Operadores usados					Cálculo Frequência		Classificação			Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X					X		Decision Tree	Cross-Validation		Erro de memória		
X					X		Maximum Entropy (algoritmo iis)	Cross-Validation				
X					X		Maximum Entropy (algoritmo gis)	Cross-Validation				
X					X		Logistic Regression	Cross-Validation		Pos – 0.863 Neg – 0.840	0.85	Pos – 0.842 Neg – 0.859
X					X		NB Bernoulli	Cross-Validation		Pos – 0.819 Neg – 0.867	0.843	Pos – 0.860 Neg – 0.828
X					X		NB Multinomial	Cross-Validation		Pos – 0.821 Neg – 0.870	0.844	Pos – 0.863 Neg – 0.829

X					X		Linear SVC (SVM)	Cross-Validation	Pos – 0.872 Neg – 0.849	0.861	Pos – 0.852 Neg – 0.870
X					X		Nu SVC (SVM)	Cross-Validation	Pos – 0.886 Neg – 0.834	0.861	Pos – 0.842 Neg – 0.881
X					X		SVC (SVM)	Cross-Validation	Pos – 0.5 Neg – 0.5	0.479	Pos - Neg -

Tabela 4 - TF-IDF sem operadores - Cross-Validation

1.1.2.2 Com operadores

Operadores usados					**Cálculo Frequência**		**Classificação**			**Valores obtidos**		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X	X	X	X		X		Decision Tree	Cross-Validation		Erro de memória		
X	X	X	X		X		Maximum Entropy (algoritmo iis)	Cross-Validation				

X	X	X	X		X		Maximum Entropy (algoritmo gis)	Cross-Validation			
X	X	X	X		X		Logistic Regression	Cross-Validation	Pos – 0.863 Neg – 0.824	0.843	Pos – 0.83 Neg – 0.857
X	X	X	X		X		NB Bernoulli	Cross-Validation	Pos – 0.804 Neg – 0.858	0.841	Pos – 0.850 Neg – 0.813
X	X	X	X		X		NB Multinomial	Cross-Validation	Pos – 0.844 Neg – 0.844	0.843	Pos – 0.844 Neg – 0.844
X	X	X	X		X		Linear SVC (SVM)	Cross-Validation	Pos – 0.867 Neg – 0.832	0.849	Pos – 0.836 Neg – 0.862
X	X	X	X		X		Nu SVC (SVM)	Cross-Validation	Pos – 0.878 Neg – 0.825	0.851	Pos – 0.833 Neg – 0.871
X	X	X	X		X		SVC (SVM)	Cross-Validation	Pos – 0.5 Neg – 0.5	0.479	Pos - Neg -

Tabela 5 – TF-IDF com operadores - Cross-Validation

1.1.2.3 Com operadores e Prune percentual (<3%)

Operadores usados					Cálculo Frequência		Classificação			Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X	X	X	X	3%	X		Decision Tree	Cross-Validation		Erro de memória		
X	X	X	X	3%	X		Maximum Entropy (algoritmo iis)	Cross-Validation				
X	X	X	X	3%	X		Maximum Entropy (algoritmo gis)	Cross-Validation				
X	X	X	X	3%	X		Logistic Regression	Cross-Validation		Pos – 0.854 Neg – 0.849	0.852	Pos – 0.849 Neg – 0.854
X	X	X	X	3%	X		NB Bernoulli	Cross-Validation		Pos – 0.766 Neg – 0.849	0.808	Pos – 0.835 Neg – 0.784
X	X	X	X	3%	X		NB Multinomial	Cross-Validation		Pos – 0.810 Neg – 0.830	0.82	Pos – 0.827 Neg – 0.814
X	X	X	X	3%	X		Linear SVC (SVM)	Cross-Validation		Pos – 0.849 Neg – 0.838	0.844	Pos – 0.840 Neg – 0.849

X	X	X	X	3%	X		Nu SVC (SVM)	Cross-Validation	Pos – 0.856 Neg – 0.837	0.847	Pos – 0.840 Neg – 0.855
X	X	X	X	3%	X		SVC (SVM)	Cross-Validation	Pos – 0.5 Neg – 0.5	0.479	Pos - Neg -

Tabela 6 – TD-IDF com operadores e Prune - Cross-Validation

1.2 70% treino – 30% teste

1.2.1 Binário

1.2.1.1 Sem operadores

Operadores usados					Cálculo Frequência		Classificação			Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X						X	Decision Tree	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.777 Neg - 0.603	0.69	Pos – 0.662 Neg – 0.729
X						X	Maximum Entropy (algoritmo iis)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Erro de memória		
X						X	Maximum Entropy (algoritmo gis)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.983 Neg – 0.463	0.723	Pos – 0.647 Neg – 0.965
X						X	Logistic Regression	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.86 Neg – 0.907	0.883	Pos – 0.902 Neg – 0.866

X						X	NB Bernoulli	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.68 Neg – 0.913	0.797	Pos – 0.887 Neg – 0.74
X						X	NB Multinomial	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.77 Neg – 0.873	0.822	Pos – 0.859 Neg – 0.791
X						X	Linear SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.853 Neg – 0.88	0.867	Pos – 0.877 Neg – 0.857
X						X	Nu SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.86 Neg – 0.9	0.88	Pos – 0.896 Neg – 0.865
X						X	SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.397 Neg – 0.97	0.683	Pos – 0.930 Neg – 0.616

Tabela 7 – Binário sem operadores – 70% / 30%

1.2.1.2 Com operadores

Operadores usados					Cálculo Frequência		Classificação			Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X	X	X	X			X	Decision Tree	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.707 Neg – 0.613	0.66	Pos – 0.646 Neg – 0.676
X	X	X	X			X	Maximum Entropy (algoritmo iis)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 1 Neg – 0	0.5	Pos – 0.5 Neg –
X	X	X	X			X	Maximum Entropy (algoritmo gis)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.983 Neg – 0.383	0.683	Pos – 0.615 Neg – 0.958
X	X	X	X			X	Logistic Regression	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.843 Neg – 0.883	0.863	Pos – 0.878 Neg -0.849

X	X	X	X			X	NB Bernoulli	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.687 Neg – 0.88	0.783	Pos – 0.851 Neg – 0.737
X	X	X	X			X	NB Multinomial	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.81 Neg – 0.833	0.822	Pos – 0.829 Neg – 0.814
X	X	X	X			X	Linear SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.833 Neg – 0.873	0.853	Pos – 0.868 Neg – 0.840
X	X	X	X			X	Nu SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.85 Neg – 0.86	0.855	Pos – 0.859 Neg – 0.851
X	X	X	X			X	SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.313 Neg – 0.97	0.642	Pos – 0.913 Neg – 0.585

Tabela 8 – Binário com operadores – 70% / 30%

1.2.1.3 Com operadores e Prune percentual (<3%)

Operadores usados					Cálculo Frequência		Classificação			Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X	X	X	X	3%		X	Decision Tree	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.69 Neg – 0.64	0.665	Pos – 0.657 Neg – 0.674
X	X	X	X	3%		X	Maximum Entropy (algoritmo iis)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.653 Neg – 0.647	0.65	Pos – 0.649 Neg – 0.651
X	X	X	X	3%		X	Maximum Entropy (algoritmo gis)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.993 Neg – 0.27	0.632	Pos – 0.576 Neg – 0.976
X	X	X	X	3%		X	Logistic Regression	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.827 Neg – 0.82	0.823	Pos – 0.821 Neg - 0.825
X	X	X	X	3%		X	NB	(0,700);	(700,1000);	Pos – 0.757	0.792	Pos – 0.814

							Bernoulli	(1000,1700)	(1700,2000)	Neg – 0.827		Neg – 0.773
X	X	X	X	3%		X	NB Multinomial	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.807 Neg – 0.827	0.817	Pos – 0.823 Neg – 0.81
X	X	X	X	3%		X	Linear SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.8 Neg – 0.783	0.792	Pos – 0.787 Neg – 0.797
X	X	X	X	3%		X	Nu SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.827 Neg – 0.83	0.828	Pos – 0.829 Neg – 0.827
X	X	X	X	3%		X	SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.763 Neg – 0.883	0.823	Pos – 0.867 Neg – 0.789

Tabela 9 – Binário com operadores e Prune – 70% / 30%

1.2.2 TF-IDF

1.2.2.1 Sem operadores

Operadores usados					Cálculo Frequência		Classificação			Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X					X		Decision Tree	(0,700); (1000,1700)	(700,1000); (1700,2000)	Erro de memória		
X					X		Maximum Entropy (algoritmo iis)	(0,700); (1000,1700)	(700,1000); (1700,2000)			
X					X		Maximum Entropy (algoritmo gis)	(0,700); (1000,1700)	(700,1000); (1700,2000)			
X					X		Logistic Regression	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.853 Neg – 0.84	0.847	Pos – 0.842 Neg – 0.851
X					X		NB	(0,700);	(700,1000);	Pos – 0.81	0.83	Pos – 0.844

							Bernoulli	(1000,1700)	(1700,2000)	Neg – 0.85		Neg – 0.817
X					X		NB Multinomial	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.82 Neg – 0.86	0.84	Pos – 0.854 Neg – 0.827
X					X		Linear SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.847 Neg – 0.857	0.852	Pos – 0.855 Neg – 0.848
X					X		Nu SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.87 Neg - 0.82	0.845	Pos – 0.829 Neg – 0.863
X					X		SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.927 Neg – 0.713	0.82	Pos – 0.764 Neg – 0.907

Tabela 10 - TF-IDF sem operadores - 70% / 30%

1.2.2.2 Com operadores

Operadores usados					**Cálculo Frequência**		**Classificação**			**Valores obtidos**		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**

X	X	X	X		X		Decision Tree	(0,700); (1000,1700)	(700,1000); (1700,2000)	Erro de memória			
X	X	X	X		X		Maximum Entropy (algoritmo iis)	(0,700); (1000,1700)	(700,1000); (1700,2000)				
X	X	X	X		X		Maximum Entropy (algoritmo gis)	(0,700); (1000,1700)	(700,1000); (1700,2000)				
X	X	X	X		X		Logistic Regression	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.837 Neg – 0.84	0.838	Pos – 0.839 Neg – 0.837	
X	X	X	X		X		NB Bernoulli	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.81 Neg – 0.87	0.84	Pos – 0.862 Neg – 0.821	
X	X	X	X		X		NB Multinomial	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.83 Neg – 0.857	0.843	Pos – 0.853 Neg – 0.834	
X	X	X	X		X		Linear SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.82 Neg – 0.843	0.832	Pos – 0.840 Neg – 0.824	
X	X	X	X		X		Nu SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.87 Neg -0.81	0.84	Pos – 0.821 Neg – 0.862	

X	X	X	X		X		SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.89 Neg – 0.787	0.838	Pos – 0.807 Neg – 0.877

Tabela 11 - TF-IDF com operadores - 70% / 30%

1.2.2.3 Com operadores e Prune percentual (<3%)

Operadores usados					Cálculo Frequência		Classificação			Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Binário**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
X	X	X	X	3%	X		Decision Tree	(0,700); (1000,1700)	(700,1000); (1700,2000)	Erro de memória		
X	X	X	X	3%	X		Maximum Entropy (algoritmo iis)	(0,700); (1000,1700)	(700,1000); (1700,2000)			
X	X	X	X	3%	X		Maximum Entropy (algoritmo	(0,700); (1000,1700)	(700,1000); (1700,2000)			

							gis)					
X	X	X	X	3%	X		Logistic Regression	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.847 Neg – 0.833	0.84	Pos – 0.835 Neg – 0.845
X	X	X	X	3%	X		NB Bernoulli	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.777 Neg – 0.84	0.808	Pos – 0.829 Neg – 0.790
X	X	X	X	3%	X		NB Multinomial	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.807 Neg – 0.84	0.823	Pos – 0.834 Neg – 0.813
X	X	X	X	3%	X		Linear SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.853 Neg – 0.797	0.825	Pos – 0.808 Neg – 0.844
X	X	X	X	3%	X		Nu SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.857 Neg – 0.807	0.832	Pos – 0.816 Neg – 0.849
X	X	X	X	3%	X		SVC (SVM)	(0,700); (1000,1700)	(700,1000); (1700,2000)	Pos – 0.81 Neg – 0.83	0.82	Pos – 0.826 Neg – 0.814

Tabela 12 – TF-IDF com operadores e Prune – 70% / 30%

Classificação de Documentos com Processamento de Linguagem Natural

Anexo F – Resultados da classificação de documentos com Python – 2ª versão (Movie Reviews)

Índice

Índice de Tabelas

Introdução

Este documento tem como objetivo divulgar os resultados da classificação de documentos com a segunda versão do código em Python. O *dataset* para treino foi o seguinte: NLTK Movie Reviews Corpus. Este conjunto de documentos é relativo à crítica de filmes e é composto por 2000 documentos divididos em duas classes: crítica positiva (1000 documentos) e crítica negativa (1000 documentos).

1 Resultados

1.1 Naive Bayes (nltk)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snow ball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	Naïve Bayes (nltk)	70% / 30%	Pos – 0.97 Neg – 0.47	0.72	Pos – 0.647 Neg – 0.94
X				X (TF-IDF)			X		Naïve Bayes (nltk)	70% / 30%	Pos – 0.82 Neg – 0.86	0.84	Pos – 0.854 Neg – 0.827
X					10000		X		Naïve Bayes (nltk)	70% / 30%	Pos – 0.983 Neg – 0.873	0.928	Pos – 0.886 Neg – 0.981

X					10000	200	X		Naïve Bayes (nltk)	70% / 30%	Pos – 0.89 Neg – 0.913	0.902	Pos – 0.911 Neg – 0.929
X				X				X	Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.969 Neg – 0.467	0.718	Pos – 0.650 Neg – 0.938
X				X (TF-IDF)			X		Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.814 Neg – 0.885	0.850	Pos – 0.876 Neg – 0.827
X					10000		X		Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.978 Neg – 0.867	0.923	Pos – 0.881 Neg – 0.975
X					10000	200	X		Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.922 Neg – 0.895	0.909	Pos – 0.898 Neg – 0.92
X	X	X		X				X	Naïve Bayes (nltk)	70% / 30%	Pos – 0.973 Neg – 0.457	0.715	Pos – 0.642 Neg – 0.945
X	X	X		X (TF-IDF)			X		Naïve Bayes (nltk)	70% / 30%	Pos – 0.83 Neg – 0.857	0.843	Pos – 0.853 Neg – 0.834
X	X	X			10000		X		Naïve Bayes (nltk)	70% / 30%	Pos – 0.977 Neg – 0.873	0.925	Pos – 0.885 Neg – 0.974
X	X	X			10000	200	X		Naïve Bayes (nltk)	70% / 30%	Pos – 0.95 Neg – 0.877	0.913	Pos – 0.885 Neg – 0.946

X	X	X		X				X	Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.972 Neg – 0.452	0.712	Pos – 0.494 Neg – 0.942
X	X	X		X (TF-IDF)			X		Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.82 Neg – 0.881	0.851	Pos – 0.874 Neg – 0.831
X	X	X			10000		X		Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.975 Neg – 0.862	0.919	Pos – 0.877 Neg – 0.972
X	X	X			10000	200	X		Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.941 Neg – 0.873	0.907	Pos – 0.882 Neg – 0.937
X	X	X	X	X				X	Naïve Bayes (nltk)	70% / 30%	Pos – 0.977 Neg – 0.407	0.692	Pos – 0.622 Neg – 0.946
X	X	X	X	X (TF-IDF)			X		Naïve Bayes (nltk)	70% / 30%	Pos – 0.827 Neg – 0.853	0.84	Pos – 0.849 Neg – 0.831
X	X	X	X		10000		X		Naïve Bayes (nltk)	70% / 30%	Pos – 0.98 Neg – 0.75	0.865	Pos – 0.797 Neg – 0.974
X	X	X	X		10000	200	X		Naïve Bayes (nltk)	70% / 30%	Pos – 0.75 Neg – 0.83	0.79	Pos – 0.815 Neg – 0.768
X	X	X	X	X				X	Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.976 Neg – 0.42	0.698	Pos – 0.631 Neg – 0.944

X	X	X	X	X (TF-IDF)			X		Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.809 Neg – 0.855	0.832	Pos – 0.848 Neg – 0.817
X	X	X	X		10000		X		Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.976 Neg – 0.744	0.86	Pos – 0.794 Neg – 0.969
X	X	X	X		10000	200	X		Naïve Bayes (nltk)	C-V (k=4)	Pos – 0.735 Neg – 0.798	0.767	Pos – 0.785 Neg – 0.751

Tabela 1 – Naïve Bayes (nltk)

1.2 Multinomial NB (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	Multinomial NB	70% / 30%	Pos – 0.77 Neg – 0.873	0.822	Pos – 0.859 Neg – 0.791
X				X (TF-IDF)			X		Multinomial NB	70% / 30%	Pos – 0.82 Neg – 0.86	0.84	Pos – 0.854 Neg – 0.827
X					10000		X		Multinomial NB	70% / 30%	Pos – 0.91 Neg – 0.94	0.925	Pos – 0.938 Neg – 0.913
X					10000	200	X		Multinomial NB	70% / 30%	Pos – 0.92 Neg – 0.927	0.923	Pos – 0.926 Neg – 0.920
X				X				X	Multinomial NB	C-V (k=4)	Pos – 0.784 Neg – 0.852	0.818	Pos – 0.842 Neg – 0.798

X				X (TF-IDF)			X		Multinomial NB	C-V (k=4)	Pos – 0.814 Neg – 0.885	0.850	Pos – 0.876 Neg – 0.827
X					10000		X		Multinomial NB	C-V (k=4)	Pos – 0.921 Neg – 0.925	0.923	Pos – 0.925 Neg – 0.922
X					10000	200	X		Multinomial NB	C-V (k=4)	Pos – 0.908 Neg – 0.912	0.91	Pos – 0.912 Neg – 0.909
X	X	X		X				X	Multinomial NB	70% / 30%	Pos – 0.787 Neg – 0.85	0.818	Pos – 0.840 Neg – 0.799
X	X	X		X (TF-IDF)			X		Multinomial NB	70% / 30%	Pos – 0.83 Neg – 0.857	0.843	Pos – 0.853 Neg – 0.834
X	X	X			10000		X		Multinomial NB	70% / 30%	Pos – 0.92 Neg – 0.94	0.93	Pos – 0.939 Neg – 0.921
X	X	X			10000	200	X		Multinomial NB	70% / 30%	Pos – 0.943 Neg – 0.883	0.913	Pos – 0.890 Neg – 0.940
X	X	X		X				X	Multinomial NB	C-V (k=4)	Pos – 0.81 Neg – 0.833	0.822	Pos – 0.830 Neg – 0.815
X	X	X		X (TF-IDF)			X		Multinomial NB	C-V (k=4)	Pos – 0.82 Neg – 0.881	0.851	Pos – 0.874 Neg – 0.831

X	X	X			10000		X		Multinomial NB	C-V (k=4)	Pos – 0.929 Neg – 0.921	0.925	Pos – 0.922 Neg – 0.904
X	X	X			10000	200	X		Multinomial NB	C-V (k=4)	Pos – 0.936 Neg – 0.882	0.909	Pos – 0.889 Neg – 0.932
X	X	X	X	X				X	Multinomial NB	70% / 30%	Pos – 0.81 Neg – 0.837	0.823	Pos – 0.832 Neg – 0.815
X	X	X	X	X (TF-IDF)			X		Multinomial NB	70% / 30%	Pos – 0.827 Neg – 0.853	0.84	Pos – 0.849 Neg – 0.831
X	X	X	X		10000		X		Multinomial NB	70% / 30%	Pos – 0.887 Neg – 0.917	0.902	Pos – 0.914 Neg – 0.890
X	X	X	X		10000	200	X		Multinomial NB	70% / 30%	Pos – 0.73 Neg – 0.843	0.787	Pos – 0.823 Neg – 0.757
X	X	X	X	X				X	Multinomial NB	C-V (k=4)	Pos – 0.81 Neg – 0.829	0.820	Pos – 0.826 Neg – 0.814
X	X	X	X	X (TF-IDF)			X		Multinomial NB	C-V (k=4)	Pos – 0.809 Neg – 0.855	0.832	Pos – 0.848 Neg – 0.817
X	X	X	X		10000		X		Multinomial NB	C-V (k=4)	Pos – 0.882 Neg – 0.908	0.895	Pos – 0.906 Neg – 0.886

X	X	X	X		10000	200	X		Multinomial NB	C-V (k=4)	Pos – 0.728 Neg – 0.817	0.773	Pos – 0.800 Neg – 0.75

Tabela 2 - Multinomial NB (Sk-learn)

1.3 Bernoulli NB (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	Bernoulli NB	70% / 30%	Pos – 0.68 Neg – 0.913	0.797	Pos – 0.887 Neg – 0.74
X				X (TF-IDF)			X		Bernoulli NB	70% / 30%	Pos – 0.81 Neg – 0.85	0.83	Pos – 0.844 Neg – 0.817
X					10000		X		Bernoulli NB	70% / 30%	Pos – 0.897 Neg – 0.95	0.923	Pos – 0.947 Neg – 0.902
X					10000	200	X		Bernoulli NB	70% / 30%	Pos – 0.86 Neg – 0.957	0.908	Pos – 0.952 Neg – 0.872
X				X				X	Bernoulli NB	C-V (k=4)	Pos – 0.681 Neg – 0.897	0.789	Pos – 0.870 Neg – 0.738

X				X (TF-IDF)			X		Bernoulli NB	C-V (k=4)	Pos – 0.81 Neg – 0.866	0.838	Pos – 0.858 Neg – 0.821
X					10000		X		Bernoulli NB	C-V (k=4)	Pos – 0.886 Neg – 0.937	0.912	Pos – 0.934 Neg – 0.893
X					10000	200	X		Bernoulli NB	C-V (k=4)	Pos – 0.861 Neg – 0.937	0.899	Pos – 0.932 Neg – 0.872
X	X	X		X				X	Bernoulli NB	70% / 30%	Pos – 0.683 Neg – 0.917	0.8	Pos – 0.891 Neg – 0.743
X	X	X		X (TF-IDF)			X		Bernoulli NB	70% / 30%	Pos – 0.823 Neg – 0.853	0.838	Pos – 0.849 Neg – 0.828
X	X	X			10000		X		Bernoulli NB	70% / 30%	Pos – 0.903 Neg – 0.953	0.928	Pos – 0.951 Neg – 0.908
X	X	X			10000	200	X		Bernoulli NB	70% / 30%	Pos – 0.843 Neg – 0.96	0.902	Pos – 0.955 Neg – 0.860
X	X	X		X				X	Bernoulli NB	C-V (k=4)	Pos – 0.671 Neg – 0.893	0.782	Pos – 0.863 Neg – 0.731
X	X	X		X (TF-IDF)			X		Bernoulli NB	C-V (k=4)	Pos – 0.807 Neg – 0.87	0.839	Pos – 0.861 Neg – 0.819

X	X	X			10000		X		Bernoulli NB	C-V (k=4)	Pos – 0.889 Neg – 0.939	0.914	Pos – 0.936 Neg – 0.896
X	X	X			10000	200	X		Bernoulli NB	C-V (k=4)	Pos – 0.849 Neg – 0.938	0.894	Pos – 0.933 Neg – 0.863
X	X	X	X	X				X	Bernoulli NB	70% / 30%	Pos – 0.697 Neg – 0.89	0.793	Pos – 0.864 Neg – 0.746
X	X	X	X	X (TF-IDF)			X		Bernoulli NB	70% / 30%	Pos – 0.803 Neg – 0.863	0.833	Pos – 0.855 Neg – 0.814
X	X	X	X		10000		X		Bernoulli NB	70% / 30%	Pos – 0.827 Neg – 0.94	0.883	Pos – 0.932 Neg – 0.844
X	X	X	X		10000	200	X		Bernoulli NB	70% / 30%	Pos – 0.65 Neg – 0.907	0.778	Pos – 0.874 Neg – 0.721
X	X	X	X	X				X	Bernoulli NB	C-V (k=4)	Pos – 0.693 Neg – 0.878	0.786	Pos – 0.851 Neg – 0.742
X	X	X	X	X (TF-IDF)			X		Bernoulli NB	C-V (k=4)	Pos – 0.804 Neg – 0.856	0.83	Pos – 0.849 Neg – 0.815
X	X	X	X		10000		X		Bernoulli NB	C-V (k=4)	Pos – 0.813 Neg – 0.928	0.871	Pos – 0.919 Neg – 0.834

X	X	X	X		10000	200	X		Bernoulli NB	C-V (k=4)	Pos – 0.648 Neg – 0.854	0.751	Pos – 0.818 Neg – 0.708

Tabela 3 – Bernoulli NB (Sk-learn)

1.4 LinearSVC (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	LinearSVC	70% / 30%	Pos – 0.853 Neg – 0.88	0.867	Pos – 0.877 Neg – 0.857
X				X (TF-IDF)			X		LinearSVC	70% / 30%	Pos – 0.847 Neg – 0.857	0.852	Pos – 0.855 Neg – 0.848
X					10000		X		LinearSVC	70% / 30%	Pos – 0.887 Neg – 0.903	0.895	Pos – 0.902 Neg – 0.888
X					10000	200	X		LinearSVC	70% / 30%	Pos – 0.893 Neg – 0.927	0.91	Pos – 0.924 Neg – 0.897
X				X				X	LinearSVC	C-V (k=4)	Pos – 0.851 Neg – 0.863	0.857	Pos – 0.862 Neg – 0.853

X				X (TF-IDF)			X		LinearSVC	C-V (k=4)	Pos – 0.868 Neg – 0.856	0.862	Pos – 0.858 Neg – 0.867
X					10000		X		LinearSVC	C-V (k=4)	Pos – 0.874 Neg – 0.889	0.882	Pos – 0.888 Neg – 0.876
X					10000	200	X		LinearSVC	C-V (k=4)	Pos – 0.9 Neg – 0.912	0.900	Pos – 0.91 Neg – 0.89
X	X	X		X				X	LinearSVC	70% / 30%	Pos – 0.853 Neg – 0.89	0.872	Pos – 0.886 Neg – 0.859
X	X	X		X (TF-IDF)			X		LinearSVC	70% / 30%	Pos – 0.843 Neg – 0.863	0.853	Pos – 0.861 Neg – 0.846
X	X	X			10000		X		LinearSVC	70% / 30%	Pos – 0.873 Neg – 0.913	0.893	Pos – 0.910 Neg – 0.878
X	X	X			10000	200	X		LinearSVC	70% / 30%	Pos – 0.887 Neg – 0.903	0.895	Pos – 0.902 Neg – 0.889
X	X	X		X				X	LinearSVC	C-V (k=4)	Pos – 0.838 Neg – 0.863	0.851	Pos – 0.86 Neg – 0.842
X	X	X		X (TF-IDF)			X		LinearSVC	C-V (k=4)	Pos – 0.866 Neg – 0.858	0.862	Pos – 0.859 Neg – 0.866

X	X	X			10000		X		LinearSVC	C-V (k=4)	Pos – 0.864 Neg – 0.887	0.876	Pos – 0.885 Neg – 0.868
X	X	X			10000	200	X		LinearSVC	C-V (k=4)	Pos – 0.877 Neg – 0.901	0.889	Pos – 0.899 Neg – 0.881
X	X	X	X	X				X	LinearSVC	70% / 30%	Pos – 0.853 Neg – 0.867	0.86	Pos – 0.865 Neg – 0.855
X	X	X	X	X (TF-IDF)			X		LinearSVC	70% / 30%	Pos – 0.827 Neg – 0.84	0.833	Pos – 0.838 Neg – 0.829
X	X	X	X		10000		X		LinearSVC	70% / 30%	Pos – 0.877 Neg – 0.86	0.868	Pos – 0.862 Neg – 0.875
X	X	X	X		10000	200	X		LinearSVC	70% / 30%	Pos – 0.707 Neg – 0.817	0.762	Pos – 0.794 Neg – 0.736
X	X	X	X	X				X	LinearSVC	C-V (k=4)	Pos – 0.831 Neg – 0.839	0.835	Pos – 0.839 Neg – 0.833
X	X	X	X	X (TF-IDF)			X		LinearSVC	C-V (k=4)	Pos – 0.85 Neg – 0.838	0.844	Pos – 0.840 Neg – 0.850
X	X	X	X		10000		X		LinearSVC	C-V (k=4)	Pos – 0.855 Neg – 0.855	0.855	Pos – 0.855 Neg – 0.856

X	X	X	X		10000	200	X		LinearSVC	C-V (k=4)	Pos – 0.72 Neg – 0.77	0.745	Pos – 0.758 Neg – 0.734

Tabela 4 – Linear SVC (Sk-learn)

1.5 NuSVC (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	NuSVC	70% / 30%	Pos – 0.86 Neg – 0.9	0.88	Pos – 0.896 Neg – 0.865
X				X (TF-IDF)			X		NuSVC	70% / 30%	Pos – 0.87 Neg – 0.82	0.845	Pos – 0.829 Neg – 0.863
X					10000		X		NuSVC	70% / 30%	Pos – 0.903 Neg – 0.907	0.905	Pos – 0.906 Neg – 0.904
X					10000	200	X		NuSVC	70% / 30%	Pos – 0.89 Neg – 0.923	0.907	Pos – 0.921 Neg – 0.893
X				X				X	NuSVC	C-V (k=4)	Pos – 0.85 Neg – 0.884	0.867	Pos – 0.881 Neg – 0.856

X				X (TF-IDF)			X		NuSVC	C-V (k=4)	Pos – 0.893 Neg – 0.812	0.853	Pos – 0.827 Neg – 0.885
X					10000		X		NuSVC	C-V (k=4)	Pos – 0.88 Neg – 0.906	0.893	Pos – 0.904 Neg – 0.884
X					10000	200	X		NuSVC	C-V (k=4)	Pos – 0.882 Neg – 0.909	0.896	Pos – 0.907 Neg – 0.886
X	X	X		X				X	NuSVC	70% / 30%	Pos – 0.847 Neg – 0.887	0.867	Pos – 0.882 Neg – 0.853
X	X	X		X (TF-IDF)			X		NuSVC	70% / 30%	Pos – 0.9 Neg – 0.82	0.86	Pos – 0.833 Neg – 0.891
X	X	X			10000		X		NuSVC	70% / 30%	Pos – 0.89 Neg – 0.9	0.895	Pos – 0.899 Neg – 0.891
X	X	X			10000	200	X		NuSVC	70% / 30%	Pos – 0.89 Neg – 0.91	0.9	Pos – 0.908 Neg – 0.892
X	X	X		X				X	NuSVC	C-V (k=4)	Pos – 0.838 Neg – 0.889	0.864	Pos – 0.883 Neg – 0.847
X	X	X		X (TF-IDF)			X		NuSVC	C-V (k=4)	Pos – 0.885 Neg – 0.803	0.844	Pos – 0.819 Neg – 0.876

X	X	X			10000		X		NuSVC	C-V (k=4)	Pos – 0.873 Neg – 0.902	0.888	Pos – 0.90 Neg – 0.877
X	X	X			10000	200	X		NuSVC	C-V (k=4)	Pos – 0.877 Neg – 0.9	0.889	Pos – 0.898 Neg – 0.881
X	X	X	X	X				X	NuSVC	70% / 30%	Pos – 0.857 Neg – 0.88	0.868	Pos – 0.877 Neg – 0.860
X	X	X	X	X (TF-IDF)			X		NuSVC	70% / 30%	Pos – 0.887 Neg – 0.763	0.825	Pos – 0.789 Neg – 0.871
X	X	X	X		10000		X		NuSVC	70% / 30%	Pos – 0.89 Neg – 0.873	0.882	Pos – 0.875 Neg – 0.888
X	X	X	X		10000	200	X		NuSVC	70% / 30%	Pos – 0.757 Neg – 0.79	0.773	Pos – 0.783 Neg – 0.765
X	X	X	X	X				X	NuSVC	C-V (k=4)	Pos – 0.831 Neg – 0.862	0.847	Pos – 0.858 Neg – 0.837
X	X	X	X	X (TF-IDF)			X		NuSVC	C-V (k=4)	Pos – 0.865 Neg – 0.813	0.839	Pos – 0.823 Neg – 0.860
X	X	X	X		10000		X		NuSVC	C-V (k=4)	Pos – 0.859 Neg – 0.874	0.867	Pos – 0.872 Neg – 0.862

X	X	X	X		10000	200	X		NuSVC	C-V (k=4)	Pos – 0.769 Neg – 0.739	0.754	Pos – 0.747 Neg – 0.762

Tabela 5 – NuSVC (Sk-learn)

1.6 LogisticRegression (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	Logistic Regression	70% / 30%	Pos – 0.86 Neg – 0.907	0.883	Pos – 0.902 Neg – 0.866
X				X (TF-IDF)			X		Logistic Regression	70% / 30%	Pos – 0.853 Neg – 0.84	0.847	Pos – 0.842 Neg – 0.851
X					10000		X		Logistic Regression	70% / 30%	Pos – 0.89 Neg – 0.913	0.902	Pos – 0.911 Neg – 0.892
X					10000	200	X		Logistic Regression	70% / 30%	Pos – 0.897 Neg – 0.923	0.91	Pos – 0.921 Neg – 0.899
X				X				X	Logistic Regression	C-V (k=4)	Pos – 0.859 Neg – 0.88	0.869	Pos – 0.879 Neg – 0.862

X				X (TF-IDF)			X		Logistic Regression	C-V (k=4)	Pos – 0.856 Neg – 0.845	0.851	Pos – 0.847 Neg – 0.855
X					10000		X		Logistic Regression	C-V (k=4)	Pos – 0.884 Neg – 0.901	0.893	Pos – 0.900 Neg – 0.886
X					10000	200	X		Logistic Regression	C-V (k=4)	Pos – 0.881 Neg – 0.907	0.894	Pos – 0.905 Neg – 0.885
X	X	X		X				X	Logistic Regression	70% / 30%	Pos – 0.86 Neg – 0.883	0.872	Pos – 0.881 Neg – 0.863
X	X	X		X (TF-IDF)			X		Logistic Regression	70% / 30%	Pos – 0.85 Neg – 0.85	0.85	Pos – 0.85 Neg – 0.85
X	X	X			10000		X		Logistic Regression	70% / 30%	Pos – 0.89 Neg – 0.91	0.90	Pos – 0.908 Neg – 0.892
X	X	X			10000	200	X		Logistic Regression	70% / 30%	Pos – 0.9 Neg – 0.903	0.902	Pos – 0.903 Neg – 0.900
X	X	X		X				X	Logistic Regression	C-V (k=4)	Pos – 0.857 Neg – 0.872	0.865	Pos – 0.871 Neg – 0.860
X	X	X		X (TF-IDF)			X		Logistic Regression	C-V (k=4)	Pos – 0.855 Neg – 0.841	0.848	Pos – 0.844 Neg – 0.854

X	X	X			10000		X		Logistic Regression	C-V (k=4)	Pos – 0.887 Neg – 0.898	0.893	Pos – 0.897 Neg – 0.889
X	X	X			10000	200	X		Logistic Regression	C-V (k=4)	Pos – 0.88 Neg – 0.893	0.887	Pos – 0.892 Neg – 0.883
X	X	X	X	X				X	Logistic Regression	70% / 30%	Pos – 0.86 Neg – 0.88	0.87	Pos – 0.878 Neg – 0.863
X	X	X	X	X (TF-IDF)			X		Logistic Regression	70% / 30%	Pos – 0.843 Neg – 0.833	0.838	Pos – 0.835 Neg – 0.842
X	X	X	X		10000		X		Logistic Regression	70% / 30%	Pos – 0.887 Neg – 0.887	0.887	Pos – 0.887 Neg – 0.887
X	X	X	X		10000	200	X		Logistic Regression	70% / 30%	Pos – 0.737 Neg – 0.797	0.767	Pos – 0.784 Neg – 0.752
X	X	X	X	X				X	Logistic Regression	C-V (k=4)	Pos – 0.848 Neg – 0.857	0.853	Pos – 0.857 Neg – 0.85
X	X	X	X	X (TF-IDF)			X		Logistic Regression	C-V (k=4)	Pos – 0.842 Neg – 0.819	0.831	Pos – 0.824 Neg – 0.839
X	X	X	X		10000		X		Logistic Regression	C-V (k=4)	Pos – 0.862 Neg – 0.867	0.865	Pos – 0.867 Neg – 0.864

X	X	X	X		10000	200	X		Logistic Regression	C-V (k=4)	Pos – 0.736 Neg – 0.759	0.748	Pos – 0.754 Neg – 0.742

Tabela 6 – Logistic Regression (Sk-learn)

Classificação de Documentos com Processamento de Linguagem Natural

Anexo G – Resultados da classificação de documentos com Python – 2ª versão (Ohsumed)

Índice

Índice de Tabelas

Introdução

Este documento tem como objetivo divulgar os resultados da classificação de documentos com a segunda versão do código em Python. O *dataset* para treino foi o seguinte: Ohsumed collection. Este conjunto de documentos é relativo a resumos de artigos de medicina e é composto por 56984 documentos divididos em 23 classes:

- **C01 Bacterial Infections and Mycoses (2540 documentos)**
- **C02 Virus Diseases (1171 documentos)**
- **C03 Parasitic Diseases (427 documentos)**
- **C04 Neoplasms (6327 documentos)**
- **C05 Musculoskeletal Diseases (1678 documentos)**
- **C06 Digestive System Diseases (2989 documentos)**
- **C07 Stomatognathic Diseases (526 documentos)**
- **C08 Respiratory Tract Diseases (2589 documentos)**
- **C09 Otorhinolaryngologic Diseases (715 documentos)**
- **C10 Nervous System Diseases (3851 documentos)**
- **C11 Eye Diseases (998 documentos)**
- **C12 Urologic and Male Genital Diseases (2518 documentos)**
- **C13 Female Genital Diseases and Pregnancy Complications (1623 documentos)**
- **C14 Cardiovascular Diseases (6102 documentos)**
- **C15 Hemic and Lymphatic Diseases (1277 documentos)**
- **C16 Neonatal Diseases and Abnormalities (1086 documentos)**
- **C17 Skin and Connective Tissue Diseases (1617 documentos)**
- **C18 Nutritional and Metabolic Diseases (1919 documentos)**
- **C19 Endocrine Diseases (865 documentos)**
- **C20 Immunologic Diseases (3116 documentos)**
- **C21 Disorders of Environmental Origin (2933 documentos)**
- **C22 Animal Diseases (506 documentos)**
- **C23 Pathological Conditions, Signs and Symptoms (9611 documentos)**

Neste caso de estudo foram criados 23 classificadores. Assim, um documento ao ser classificado terá de ser testado em cada um dos classificadores e será classificado na classe em que tiver maior valor de confiança. Todos os classificadores foram realizados com 70% treino de documentos da classe positiva, e a classe negativa contém o mesmo número de documentos de treino da classe positiva divididos pelas 22 classes negativas. Os 30% de documentos de teste seguiram o mesmo princípio do conjunto de treino.

1 Resultados

1.1 Melhores features (10000)

1.1.1 Sem operadores

Classes	Operadores usados					Cálculo Frequência	Classificação			Valores obtidos		
Classes	**Transform Cases (lower case)**	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
C01	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.936 Neg – 0.806	0.872	Pos – 0.831 Neg – 0.925
C02	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.949 Neg – 0.821	0.887	Pos – 0.850 Neg – 0.938
C03	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.922 Neg – 0.891	0.908	Pos – 0.908 Neg – 0.907
C04	X					X	NaiveBayes (nltk)	70%	30%	Erro de memória		

C05	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.825 Neg – 0.876	0.850	Pos – 0.874 Neg – 0.828
C06	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.948 Neg – 0.773	0.861	Pos – 0.810 Neg – 0.935
C07	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.728 Neg – 0.929	0.827	Pos – 0.913 Neg – 0.769
C08	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.954 Neg – 0.764	0.859	Pos – 0.803 Neg – 0.942
C09	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.888 Neg – 0.949	0.918	Pos – 0.950 Neg – 0.887
C10	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.848 Neg – 0.818	0.833	Pos – 0.825 Neg – 0.842
C11	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.823 Neg – 0.965	0.892	Pos – 0.961 Neg – 0.839
C12	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.963 Neg – 0.826	0.895	Pos – 0.848 Neg – 0.957
C13	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.953 Neg – 0.862	0.907	Pos – 0.874 Neg – 0.948

C14	X					X	NaiveBayes (nltk)	70%	30%	Erro de memória		
C15	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.956 Neg – 0.743	0.851	Pos – 0.793 Neg – 0.942
C16	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.917 Neg – 0.838	0.879	Pos – 0.857 Neg – 0.905
C17	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.807 Neg – 0.903	0.855	Pos – 0.893 Neg – 0.823
C18	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.922 Neg – 0.712	0.817	Pos – 0.763 Neg – 0.900
C19	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.946 Neg – 0.707	0.831	Pos – 0.776 Neg – 0.924
C20	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.944 Neg – 0.709	0.827	Pos – 0.766 Neg – 0.926
C21	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.867 Neg – 0.840	0.854	Pos – 0.848 Neg – 0.860
C22	X					X	NaiveBayes (nltk)	70%	30%	Pos – 1.0 Neg – 0.788	0.901	Pos – 0.844 Neg – 1.0

C23	X					X	NaiveBayes (nltk)	70%	30%	Erro de memória

Tabela 1 – Melhores features (10000)

1.2 Melhores features (10000) + Melhores features Bigrams (200)

1.2.1 Sem operadores

Classes	Operadores usados					Cálculo Frequência	Classificação			Valores obtidos		
Classes	**Transform Cases (lower case)**	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Prune (percentual)**	**TF-IDF**	**Classificador**	**Treino**	**Teste**	**Recalls**	**Accuracy**	**Precision**
C01	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.942 Neg – 0.717	0.830	Pos – 0.772 Neg – 0.925
C02	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.955 Neg – 0.730	0.846	Pos – 0.791 Neg – 0.938
C03	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.938 Neg – 0.845	0.895	Pos – 0.877 Neg – 0.921
C04	X					X	NaiveBayes (nltk)	70%	30%	Erro de memória		

C05	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.853 Neg – 0.808	0.831	Pos – 0.822 Neg – 0.841
C06	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.968 Neg – 0.590	0.781	Pos – 0.706 Neg – 0.947
C07	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.772 Neg – 0.857	0.814	Pos – 0.847 Neg – 0.786
C08	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.964 Neg – 0.632	0.799	Pos – 0.726 Neg – 0.946
C09	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.893 Neg – 0.859	0.877	Pos – 0.873 Neg – 0.881
C10	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.875 Neg – 0.772	0.823	Pos – 0.795 Neg – 0.859
C11	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.863 Neg – 0.923	0.892	Pos – 0.922 Neg – 0.866
C12	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.968 Neg – 0.743	0.856	Pos – 0.792 Neg – 0.959
C13	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.936 Neg – 0.793	0.865	Pos – 0.820 Neg – 0.925

C14	X					X	NaiveBayes (nltk)	70%	30%	Erro de memória		
C15	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.961 Neg – 0.626	0.796	Pos – 0.725 Neg – 0.940
C16	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.914 Neg – 0.789	0.853	Pos – 0.821 Neg – 0.897
C17	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.856 Neg – 0.810	0.833	Pos – 0.819 Neg – 0.848
C18	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.908 Neg – 0.678	0.794	Pos – 0.740 Neg – 0.880
C19	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.931 Neg – 0.607	0.775	Pos – 0.718 Neg – 0.891
C20	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.961 Neg – 0.670	0.817	Pos – 0.747 Neg – 0.945
C21	X					X	NaiveBayes (nltk)	70%	30%	Pos – 0.853 Neg – 0.806	0.830	Pos – 0.819 Neg – 0.843
C22	X					X	NaiveBayes (nltk)	70%	30%	Pos – 1.0 Neg – 0.735	0.877	Pos – 0.813 Neg – 1.0

C23	X					X	NaiveBayes (nltk)	70%	30%	Erro de memória

Tabela 2 – Melhores features (10000) + melhores features bigrams (200)

Classificação de Documentos com Processamento de Linguagem Natural

Anexo H – Resultados da classificação de documentos com Python – 2ª versão (Enzimas)

Índice

Índice de Tabelas

Introdução

Este documento tem como objetivo divulgar os resultados da classificação de documentos com a segunda versão do código em Python. O *dataset* para treino foi das Enzimas. Este conjunto de documentos é composto por 270 documentos divididos em duas classes: Pepsin (135 documentos) e Chymotrypsin (135 documentos).

1 Resultados

1.1 Naive Bayes (nltk)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	Naïve Bayes (nltk)	70% / 30%	Pep – 0.951 Chy – 0.976	0.963	Pep – 0.975 Chy – 0.952
X				X (TF-IDF)			X		Naïve Bayes (nltk)	70% / 30%	Pep – 0.951 Chy – 0.927	0.939	Pep – 0.929 Chy – 0.95
X					3000		X		Naïve Bayes (nltk)	70% / 30%	Pep – 0.976 Chy – 1.0	0.988	Pep – 1.0 Chy – 0.976
X					3000	100	X		Naïve Bayes (nltk)	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.956 Chy – 0.976

X	X	X		X				X	Naïve Bayes (nltk)	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X		X (TF-IDF)			X		Naïve Bayes (nltk)	70% / 30%	Pep – 0.976 Chy – 0.951	0.963	Pep – 0.952 Chy – 0.975
X	X	X			3000		X		Naïve Bayes (nltk)	70% / 30%	Pep – 0.976 Chy – 1.0	0.988	Pep – 1.0 Chy – 0.976
X	X	X			3000	100	X		Naïve Bayes (nltk)	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X	X				X	Naïve Bayes (nltk)	70% / 30%	Pep – 0.951 Chy – 0.976	0.963	Pep – 0.975 Chy – 0.953
X	X	X	X	X (TF-IDF)			X		Naïve Bayes (nltk)	70% / 30%	Pep – 0.951 Chy – 0.976	0.963	Pep – 0.975 Chy – 0.953
X	X	X	X		3000		X		Naïve Bayes (nltk)	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X		3000	100	X		Naïve Bayes (nltk)	70% / 30%	Pep – 0.951 Chy – 0.902	0.927	Pep – 0.907 Chy – 0.949

Tabela 1 – Naïve Bayes (nltk)

1.2 Multinomial NB (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	Multinomial NB	70% / 30%	Pep – 0.951 Chy – 0.976	0.963	Pep – 0.975 Chy – 0.952
X				X (TF-IDF)			X		Multinomial NB	70% / 30%	Pep – 0.951 Chy – 0.927	0.939	Pep – 0.929 Chy – 0.95
X					3000		X		Multinomial NB	70% / 30%	Pep – 0.951 Chy – 1.0	0.976	Pep – 1.0 Chy – 0.953
X					3000	100	X		Multinomial NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X		X				X	Multinomial NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976

X	X	X		X (TF-IDF)			X		Multinomial NB	70% / 30%	Pep – 0.976 Chy – 0.951	0.963	Pep – 0.952 Chy – 0.975
X	X	X			3000		X		Multinomial NB	70% / 30%	Pep – 0.976 Chy – 1.0	0.988	Pep – 1.0 Chy – 0.976
X	X	X			3000	100	X		Multinomial NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X	X				X	Multinomial NB	70% / 30%	Pep – 0.951 Chy – 0.976	0.963	Pep – 0.975 Chy – 0.952
X	X	X	X	X (TF-IDF)			X		Multinomial NB	70% / 30%	Pep – 0.951 Chy – 0.976	0.963	Pep – 0.975 Chy – 0.952
X	X	X	X		3000		X		Multinomial NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X		3000	100	X		Multinomial NB	70% / 30%	Pep – 0.951 Chy – 0.951	0.951	Pep – 0.951 Chy – 0.951

Tabela 2 - Multinomial NB (Sk-learn)

1.3 Bernoulli NB (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X				X (TF-IDF)			X		Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X					3000		X		Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X					3000	100	X		Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 1.0	0.988	Pep – 1.0 Chy – 0.976
X	X	X		X				X	Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976

X	X	X		X (TF-IDF)			X		Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X			3000		X		Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 1.0	0.988	Pep – 1.0 Chy – 0.976
X	X	X			3000	100	X		Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 1.0	0.988	Pep – 1.0 Chy – 0.976
X	X	X	X	X				X	Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 0.951	0.963	Pep – 0.952 Chy – 0.975
X	X	X	X	X (TF-IDF)			X		Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X		3000		X		Bernoulli NB	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X		3000	100	X		Bernoulli NB	70% / 30%	Pep – 0.829 Chy – 0.951	0.890	Pep – 0.944 Chy – 0.848

Tabela 3 – Bernoulli NB (Sk-learn)

1.4 LinearSVC (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	LinearSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X				X (TF-IDF)			X		LinearSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X					3000		X		LinearSVC	70% / 30%	Pep – 1.0 Chy – 1.0	1.0	Pep – 1.0 Chy – 1.0
X					3000	100	X		LinearSVC	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X		X				X	LinearSVC	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0

X	X	X		X (TF-IDF)			X		LinearSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X			3000		X		LinearSVC	70% / 30%	Pep – 1.0 Chy – 1.0	1.0	Pep – 1.0 Chy – 1.0
X	X	X			3000	100	X		LinearSVC	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X	X	X				X	LinearSVC	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X	X	X (TF-IDF)			X		LinearSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X		3000		X		LinearSVC	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X	X		3000	100	X		LinearSVC	70% / 30%	Pep – 0.951 Chy – 0.951	0.951	Pep – 0.951 Chy – 0.951

Tabela 4 – Linear SVC (Sk-learn)

1.5 NuSVC (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	NuSVC	70% / 30%	Pep – 0.951 Chy – 0.976	0.963	Pep – 0.975 Chy – 0.952
X				X (TF-IDF)			X		NuSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X					3000		X		NuSVC	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X					3000	100	X		NuSVC	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X		X				X	NuSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976

X	X	X		X (TF-IDF)			X		NuSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X			3000		X		NuSVC	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X			3000	100	X		NuSVC	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X	X	X				X	NuSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X	X (TF-IDF)			X		NuSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X		3000		X		NuSVC	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X		3000	100	X		NuSVC	70% / 30%	Pep – 0.902 Chy – 0.976	0.939	Pep – 0.974 Chy – 0.909

Tabela 5 – NuSVC (Sk-learn)

1.6 LogisticRegression (Sk-learn)

Pré-processamento				Técnica			Cálculo Frequência		Classificação		Valores obtidos		
Transform Cases (lower case)	**Tokenize (non letters)**	**Filter Stopwords (English)**	**Stem (Snowball)**	**Todas as palavras**	**Melhores features**	**Melhores features bigrams**	**TF-IDF**	**Binário**	**Classificador**	**Tipo Treino**	**Recalls**	**Accuracy**	**Precision**
X				X				X	Logistic Regression	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X				X (TF-IDF)			X		Logistic Regression	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X					3000		X		Logistic Regression	70% / 30%	Pep – 0.976 Chy – 1.0	0.988	Pep – 1.0 Chy – 0.976
X					3000	100	X		Logistic Regression	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X		X				X	Logistic Regression	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X		X (TF-IDF)			X		Logistic	70% / 30%	Pep – 0.976	0.976	Pep – 0.976

									Regression		Chy – 0.976		Chy – 0.976
X	X	X			3000		X		Logistic Regression	70% / 30%	Pep – 1.0 Chy – 1.0	1.0	Pep – 1.0 Chy – 1.0
X	X	X			3000	100	X		Logistic Regression	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X	X	X				X	Logistic Regression	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X	X	X (TF-IDF)			X		Logistic Regression	70% / 30%	Pep – 0.976 Chy – 0.976	0.976	Pep – 0.976 Chy – 0.976
X	X	X	X		3000		X		Logistic Regression	70% / 30%	Pep – 1.0 Chy – 0.976	0.988	Pep – 0.976 Chy – 1.0
X	X	X	X		3000	100	X		Logistic Regression	70% / 30%	Pep – 0.951 Chy – 0.951	0.951	Pep – 0.951 Chy – 0.951

Tabela 6 – Logistic Regression (Sk-learn)