Aprendizagem Automática

Projeto Final

Ricardo Vieira A45871
Eduardo Marques A45977

Aprendizagem Supervisionada

Na Aprendizagem Supervisionada, o classificador é treinado usando dados onde já conhecemos as respostas desejadas. Ou seja, fornecemos um conjunto de dados que já sabemos como classificar, e treinamos o classificador para que ele aprenda a classificar dados no futuro dessa mesma forma. Podemos listar alguns dos métodos para este tipo de aprendizagem:

- ***Regressão Linear :*** Dados pares de entrada-saída, o modelo aprende uma relação para prever valores contínuos

- ***Classificação:*** Dados pares de entrada-classe, o modelo aprende a associar entradas a categorias específicas.

Aprendizagem não Supervisionada

Na Aprendizagem não Supervisionada, o classificador é treinado usando dados sem informações sobre as respostas desejadas. Neste caso, o modelo vai explorar padrões nos dados recebidos, sem orientações externas, de modo a classificar corretamente os dados. Os métodos de agrupamentos, mais conhecidos por Clustering, são amplamente usados na aprendizagem não supervisionada.

- *Clustering :* O algoritmo agrupa os dados em clusters, sem qualquer informação prévia sobre os mesmos, procurando semelhanças entre os dados. Procura assim dividir os dados em grupos, de modo a que os dados mais semelhantes entre si perteçam ao mesmo grupo. Assim, é capaz de categorizar os dados e descobrir as classes a que pertencem.

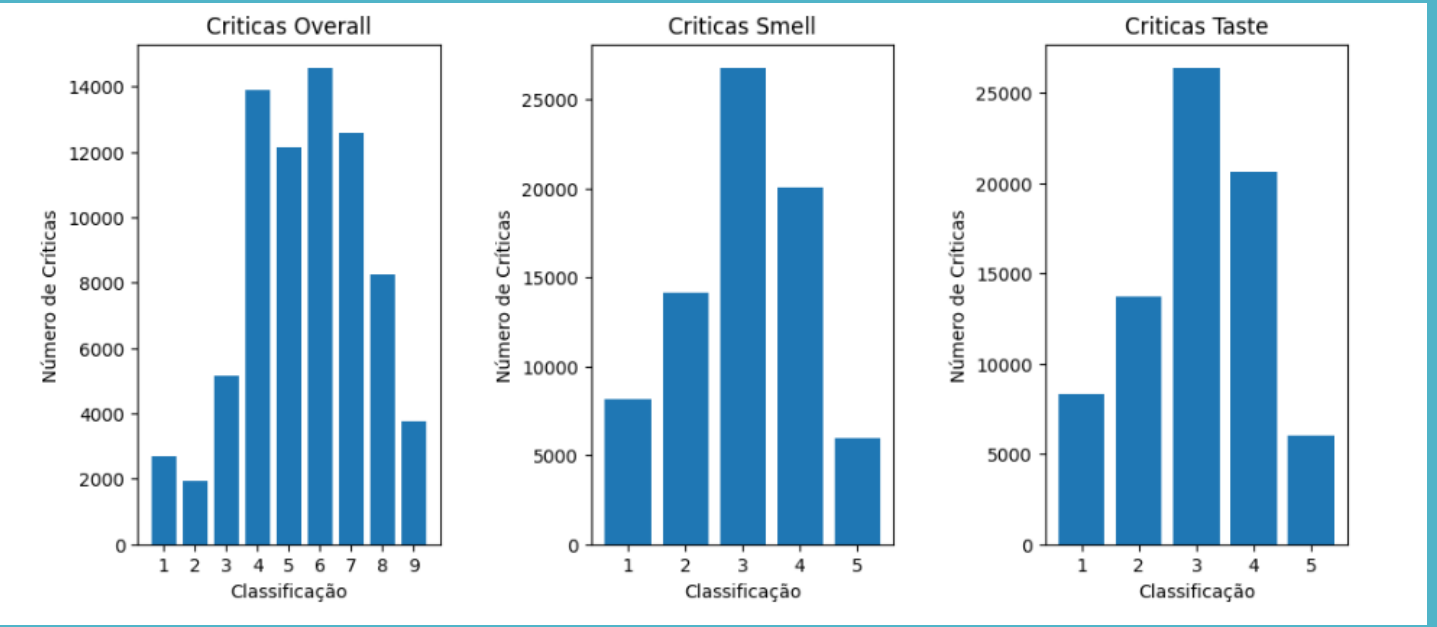
Introdução

Este trabalho tem como objetivo determinar a qualidade de uma cerveja baseado em críticas acerca desta. Para tratar este problema de classificação, usando aprendizagem supervisionada, vamos treinar vários classificadores, que deverão ser capazes de classificar os comentários de duas formas diferentes:

- ***Classificação Binária***: Tem como objetivo classificar a cerveja como muito boa ou muito má. Só considera as pontuações globais acima de 8 (muito boas) e de 2 ou menos (muito más).

- ***Classificação Multi-Classe***: Tem como objetivo classificar a cerveja em três aspetos distintos: *smell*, *taste* e *overall*. Considera todas as pontuações existentes para cada um destes tópicos.

Estudo dos dados



Estes são os dados relevantes para os nossos problemas de classificação.

Observações:

- Para as pontuações globais, a grande maioria está concentrada nas classificações centrais, entre 4 e 7. Isto significa que grande parte das críticas não será considerada para a classificação binária.
- O mesmo acontece para as críticas de *Smell* e *Taste*, onde a grande maioria das classificações tem o valor 3, que neste caso é o valor central.

Pré-Processamento

O pré-processamento dos dados é uma etapa fundamental, que consiste em manipular, limpar e transformar o conjunto de dados que vamos classificar para um formato adequado, com o objetivo de serem usados pelos classificadores mais tarde. Esta etapa tem vários passos que foram implementados no trabalho:

- *Método "cleanFile":* Este método é o primeiro passo da etapa de pré-processamento e faz o trabalho mais simples como: converter strings binárias para strings normais, remover marcas *HTML* de mudança de linha, e eliminar todos os caracteres que não sejam alfabéticos ou acentuados.

Como no nosso caso, na maioria das críticas não existem marcas HTML ou breaks, não se nota uma diferença nas críticas antes e após a aplicação da função.

Pré-Processamento

- *Stemming :* Envolve a redução das palavras para as suas formas mais básicas, removendo sufixos e prefixos. O objetivo é simplificar as palavras ao máximo para que diferentes formas da mesma palavra sejam efetivamente convetidas na mesma, reduzindo ainda a dimensão do vocabulário.

O processo de stemming pode ser realizado por diferentes algoritmos, tais como:

- *Porter Stemmer:* É o algoritmo clássico de stemming, utiliza um conjunto de regras para reduzir palavras à sua forma básica.

- *Snowball Stemmer:* É uma versão melhorada do Porter Stemmer, desenvolvida para ser mais "agressiva" na redução das palavras.

- *Lancaster Stemmer:* Outro algoritmo de stemming, também mais agressivo que o *Porter Stemmer* e *Snowball Stemmer*, resultando em palavras ainda mais curtas.

Pré-Processamento

- *TF-IDF:* A vetorização é o último passo do pré-processamento e também o mais complexo de todos. Consiste em avaliar a importância de uma palavra dentro de um dado conjunto. Alguns conceitos importantes são:

- ***Frequência da palavra no Texto (TF) :*** Se uma palavra aparece muitas vezes no texto, tem um *TF* alto para esse texto.

- ***Raridade da palavra no Conjunto (IDF) :*** Se uma palavra é rara em todo o conjunto de textos, tem um *IDF* alto.

O valor final de *TF-IDF* é uma combinação dessas duas medidas. Palavras com valores de *TF-IDF* mais altos são consideradas mais importantes e específicas para um texto. Existem ainda parâmetros importantes: *min_df* que define a quantidade de vezes que uma palavra deve aparecer para ser incluída, *token_pattern* que define o padrão que uma palavra deve seguir para ser considerada, e ainda gama de *n gramas*, que representam os tamanhos de janela de visualização de palavras.

Pré-Processamento

Para obtermos os melhores parêmetros temos de realizar testes com diferentes combinações e ver com qual delas obtemos melhores resultados. A função "***process_tfidf_vectorizer"*** vai criar representações vetorias dos documentos recebidos usando a abordagem *TF_IDF* que atribui pesos a palavras com base na sua frequência e na sua raridade em todo o conjunto.

Além do que já foi mencionado anteriormente, queremos remover todas as ***Stop Words*** que encontramos. Estas são as palavras mais comuns como "e", "o", "a", e são palavras que pouco ou nada contribuem para o vocabulário, pelo que não nos são úteis e estão apenas a aumentar a dimensão do vocabulário.

Após realizarmos uma análise do impacto dos parâmetros, podemos verificar que não existe uma diferença significativa com e sem *Stop words*, pelo que decidimos usar as criticas sem *Stop Words*. Além disso verificamos ainda existe uma maior variação de *min_df* para valores mais baixos, o que indica que há várias palavras que só apareçem uma ou duas vezes.

```
numero_palavras_sem_stop.p
[[44018 43384 40899]
 [21789 21275 19775]
 [16458 16002 14842]]

numero_palavras_com_stop.p
[[43792 43188 40744]
 [21572 21088 19629]
 [16244 15818 14698]]
```

Classificadores

Agora que temos os dados finalmente limpos, podemos passar à classificação dos mesmos. Para isso vamos usar 3 classificadores distintos, que vão obter resultados diferentes dadas as suas características.

Regressão Logística

Este método de classificação baseia-se em observações anteriores para prever a probabilidade de uma instância pertencer a uma classe com base na análise da relação entre uma ou mais variáveis.

Tem como principais vantagens a facilidade de implementação e compreensão, além de obter bons resultados quando o conjunto de dados é linearmente separável, sendo por isso ideal para problemas de classificação binária.

A principal desvantagem prende-se com o facto de ser limitado precisamente para problemas lineares, e no mundo real não é muito comum os dados serem linearmente separáveis. É também limitado a problemas que usem dados com números discretos.

Classificadores

Classificador SVM

Este tipo de classificador pode ser usado tanto na regressão como na classificação e procura um hiperplano de separação que maximize a margem entre as classes do conjunto de dados.

Para isso utiliza vetores de suporte, que correspondem aos pontos de dados mais próximos do hiperplano de separação. Esses vetores são cruciais para a definição da margem de decisão e, consequentemente, para a classificação.

Tem como vantagens a obtenção de bons resultados para problemas de classificação e para problemas com dados de altas dimensões.

A grande desvantagem está relacionada com a sua sensibilidade ao escalamento dos dados e à escolha dos parâmetros. É importante por isso fazer uma escolha correta dos parâmetros e ter também especial atenção à escala de todas as características para evitar problemas na classificação.

Classificadores

Classificador Naive Bayes

Este classificador, que é baseado no *Teorema de Bayes*, é capaz de prever a "etiqueta" de um texto, e é especialmente útil em problemas de classificação de texto.

Este teorema descreve a probabilidade condicional de um evento, dado outro evento que ocorreu. ***Multinomial Naive Bayes*** calcula a probabilidade de uma etiqueta para um conjunto de dados e depois dá a etiqueta que tiver maior probabilidade de saída.

Como vantagens temos a sua simplicidade e facilidade de implementação, assim como a sua escalabilidade, sendo capaz de lidar com conjuntos de dados com muitas características (como problemas de classificação de textos).

As desvantagens estão relacionadas com a sua sensiblilidade a características irrelevates, uma vez que este assume independência entre elas, e ainda à sua precisão de previsão ser menor quando comparada com outros algoritmos de probabilidade.

Calibração dos modelos de classificação binários

Regressão Logística

- Temos de encontrar os melhores valores para os parâmetros de *Penalty* e *Regularização*.
- Escolhemos a melhor combinação de valores através do *score* obtido.

SVM

- Temos de encontrar os melhores valores para os parâmetros de *Penalty* e *Regularização*.
- Escolhemos a melhor combinação de valores através do *score* de validação cruzada.

Naive Bayes

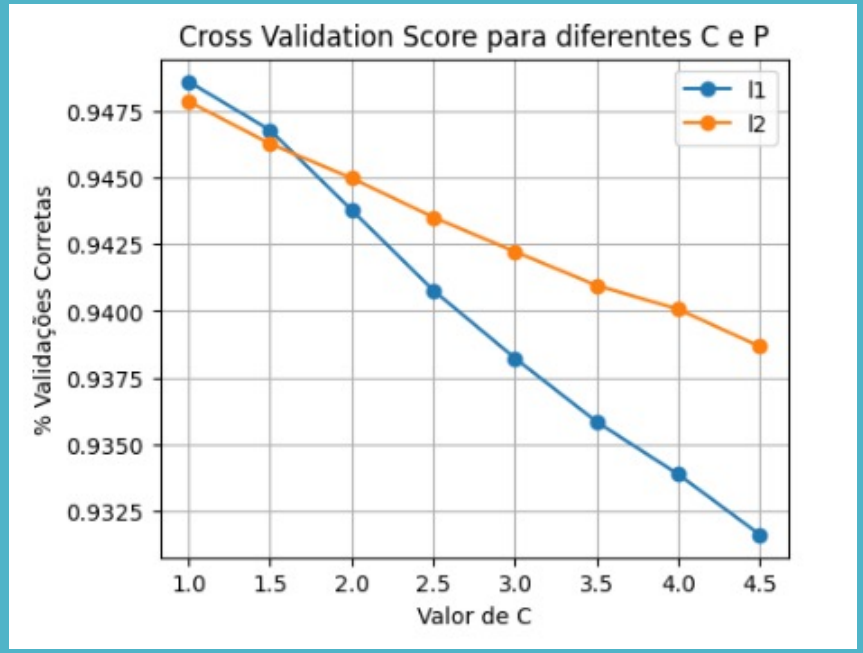
- Encontrar o melhor valor para *alpha*.
- Escolhemos a melhor combinação de valores através do *score* obtido

Implementação - Caso Binário

Parâmetros Binários

Classificador SVM: Para encontrarmos os melhores parâmetros para este classificador o processo é o mesmo que para a *regressão logística*, onde percorremos as várias combinações possíveis e guardamos aquela que obtiver a melhor pontuação, que neste caso é obtida por validação cruzada.

Neste caso vamos ter duas penalidades diferentes, *l1* que adiciona a soma dos valores absolutos dos coeficientes ao custo da função, e *l2* que adiciona a soma dos quadrados dos coeficientes.
Aqui optámos por escolher usar *l1*, uma vez que este obtém melhores resultados para quando *C=1*, que é o valor onde se verifica o início da convergência.



Implementação - Caso Binário

Parâmetros Binários

Tal como foi referido anteriormente, antes de passarmos à classificação dos dados propriamente dita, temos de achar os melhores parâmetros, de modo a obtermos os melhores resultados possíveis.

Regressão Logística: Para a regressão logística vamos usar um *loop for* que vai percorrer as listas com diferentes valores para os vários parâmetros e calcular para todas as combinações possíveis o *score* obtido. Se o *score* obtido para uma dada combinação for superior ao melhor score até aquele momento, guarda os valores desses parâmetros como os melhores. Estes resultados são guardados num ficheiro *pickle* para serem usados depois para classificar. No nosso caso, os melhores valores encontrados para os parâmetros foram os seguintes:

```
Melhores parâmetros encontrados no teste anterior: Max iter:  1000 , C:  100  e tol:  0.0001
Score de treino:  98.005 %
```

Implementação - Caso Binário

Parâmetros Binários

Classificador Naive Bayes: Para este classificador, que será usado apenas para a classificação binária, vamos mais uma vez percorrer oas listas de valores para os parâmetros *alpha*, *force_alpha* e *fit_prior*, calculando o *score* obtido para cada combinação e guardando os valores para a melhor combinação num ficheiro *pickle*.

O parâmetro *alpha* é usado para evitar que as probabilidades condicionais sejam zero. Um valor mais alto indica uma suavização mais forte.

Os parâmetros *force_alpha* e *fit_prior* são variáveis booleanas pelo que só têm dois valores possíveis, *True* ou *False*. O primeiro indica se o parâmetro *alpha* pode tomar ou não valores positivos, e o segundo indica se as probabilidades à priori das classes são ajustadas com base nos dados de treino.

Os melhores valores que obtivemos para os parâmetros foram os seguintes:

```
Melhores parâmetros encontrados no teste anterior: Alpha:  0.0001 , force_alpha:  True  e fit_prior:  True
Score de treino:  96.159 %
```

Resultados Obtidos - Caso Binário

Depois de encontrados os melhores parâmetros para cada classificador e realizada a classificação binária para cada caso, obtemos os resultados seguintes:

Regressão Logística

```
Score treino binário bom:  97.061 %
Score teste binário bom:  95.708 %
Matriz de confusão das reviews muito boas:
 [[23833   301]
 [  772    94]]
Erros:  1073
```

```
Score treino binário mau:  96.891 %
Score teste binário mau:  89.88 %
Matriz de confusão das reviews muito más:
 [[21738   588]
 [ 1942   732]]
Erros:  2530
```

Classificador SVM

```
Score treino binário bom:  95.24 %
Score teste binário bom:  96.488 %
Matriz de confusão das reviews muito boas:
 [[24112    22]
 [  856    10]]
Erros:  878
```

```
Score treino binário mau:  95.116 %
Score teste binário mau:  90.572 %
Matriz de confusão das reviews muito más:
 [[22168   158]
 [ 2199   475]]
Erros:  2357
```

Naive Bayes

```
Score treino binário bom:  96.159 %
Score teste binário bom:  96.408 %
Matriz de confusão das reviews muito boas:
 [[24097    37]
 [  861     5]]
Erros:  898
```

```
Score treino binário bom:  95.992 %
Score teste binário bom:  89.264 %
Matriz de confusão das reviews muito boas:
 [[22164   162]
 [ 2522   152]]
Erros:  2684
```

Resultados Obtidos - Caso Binário

Observando os resultados obtidos demonstrados anteriormente podemos tirar algumas conclusões:

- A *regressão logística* é o classificador com os melhores scores de treino e teste para ambos os casos (excepto para os de teste negativos). Podemos por isso afirmar que para este tipo de dados a *regressão logística* seria o classificador mais adequado, apesar do seu tempo de execução.

- Os resultados obtidos entre os *Classificadores SVM* e *Naive Bayes* variam muito pouco em ambos os casos.

- Obtemos para todos os classificadores sempre uma quantidade de erros muito maior para os resultados classificados como "maus" do que para os "bons".

Implementação - Caso Multi-Classe

Parâmetros Multi-Classe

Na classificação multi-classe é pretendido treinar e avaliar os classificadores com três dados de treino diferentes: *smell*, *taste e overall*. Neste caso, devemos abordar cada um destes problemas de forma separada, obtendo assim resultados diferentes e uma matriz de confusão para cada um dos atributos.

Mais uma vez, o primeiro passo seria encontrar os melhores parâmetros. No entanto, tanto para a *regressão linear* como para o *classificador SVM*, iremos usar os mesmos valores que foram usados para a classificação binária, devido ao seu elevado tempo de execução.

Num cenário ideal, deviamos calcular novamente os melhores valores, onde o procedimento seria exatamente o mesmo, onde iríamos testar vários parâmetros diferentes para cada um dos atributos.

Resultados Obtidos - Caso Multi-Classe

Após realizada a classificação multi-classe para cada um dos atributos em questão, obtemos os resultados seguintes:

	Overall	Smell	Taste
Regressão Logística	Score classificação overall train: 75.119 % Score classificação overall test: 25.588 % Matriz de confusão das reviews overall: [[268 221 268 299 78 78 69 72 46] [111 143 368 431 119 56 33 12 2] [100 183 572 1090 335 257 115 33 14] [126 173 622 2130 1135 899 424 155 38] [76 40 219 1038 864 893 562 223 62] [94 37 134 701 684 1144 736 364 116] [97 15 65 379 455 804 763 455 166] [69 10 29 147 182 396 483 375 182] [57 3 13 54 67 131 204 199 138]]	Score classificação smell train: 77.316 % Score classificação smell test: 40.348 % Matriz de confusão das reviews smell: [[2114 1475 990 268 51] [974 1917 2317 655 75] [579 1630 3791 1863 260] [150 444 1799 1994 442] [48 80 286 527 271]]	Score classificação taste train: 78.923 % Score classificação taste test: 41.688 % Matriz de confusão das reviews taste: [[2172 1393 986 242 27] [931 1709 2096 592 68] [520 1518 3940 1971 231] [165 436 1894 2287 489] [46 69 315 589 314]]
Classificador SVM	Score classificação overall train: 63.495 % Score classificação overall test: 27.116 % Matriz de confusão das reviews overall: [[304 110 253 381 75 84 77 72 43] [96 80 286 620 90 56 34 11 2] [78 97 405 1460 267 237 111 34 10] [86 79 417 2707 934 932 393 123 31] [50 24 142 1265 747 954 581 176 38] [62 17 93 828 619 1242 739 329 81] [54 5 59 462 374 874 814 441 116] [35 5 24 184 160 410 576 367 112] [38 0 14 67 55 149 217 213 113]]	Score classificação smell train: 68.623 % Score classificação smell test: 42.776 % Matriz de confusão das reviews smell: [[2291 1227 1140 212 28] [902 1672 2700 616 48] [454 1248 4414 1859 148] [128 317 2010 2123 251] [46 54 310 608 194]]	Score classificação taste train: 68.881 % Score classificação taste test: 44.296 % Matriz de confusão das reviews taste: [[2387 1131 1053 229 20] [863 1388 2510 599 36] [431 1069 4573 1976 131] [145 272 2055 2494 305] [53 46 323 679 232]]

Resultados Obtidos - Caso Multi-Classe

Observando os resultados que foram demonstrados anteriormente para a classificação multi-classe, podemos tirar algumas conclusões:

- Nota-se que existe uma diferença significativa para as classificações dos dados de treino nos três aspetos a classificar, onde mais uma vez a *regressão logística* aparece como sendo o melhor classificador, apesar de ficar longe de obter resultados agradáveis.

- Para os dados de teste os resultados foram muito pouco satisfatórios para os dois classificadores e não existe uma diferença significativa entre os dois.

- Posto isto, o classificador mais correto de utilizar seria novamente o de *regressão logística*, uma vez que este se comporta bem melhor que o *classificador SVM* para os dados de treino.

Resultados Obtidos - Caso Multi-Classe

Como já vimos, os resultados obtidos estão longe de serem satisfatórios para a classificação multi-classe. Algumas razões que podem justificar isso são:

- Podem haver palavras repetidas em diferentes classificações o que faz com que exista uma alta probabilidade de erro.

- Analisando as classificações *smell* e *taste*, existem também bastantes erros, o que pode significar que uma cerveja pode ter uma boa classificação *overall* mas ter as classificações de *smell* ou *taste* mais baixas.

- Para que a classificação fosse mais correta, era importante que existisse mais consistência nas classificações, ou seja, que reviews com uma classificação alta de *overall*, também tivessem pontuações altas a *smell* e *taste*.