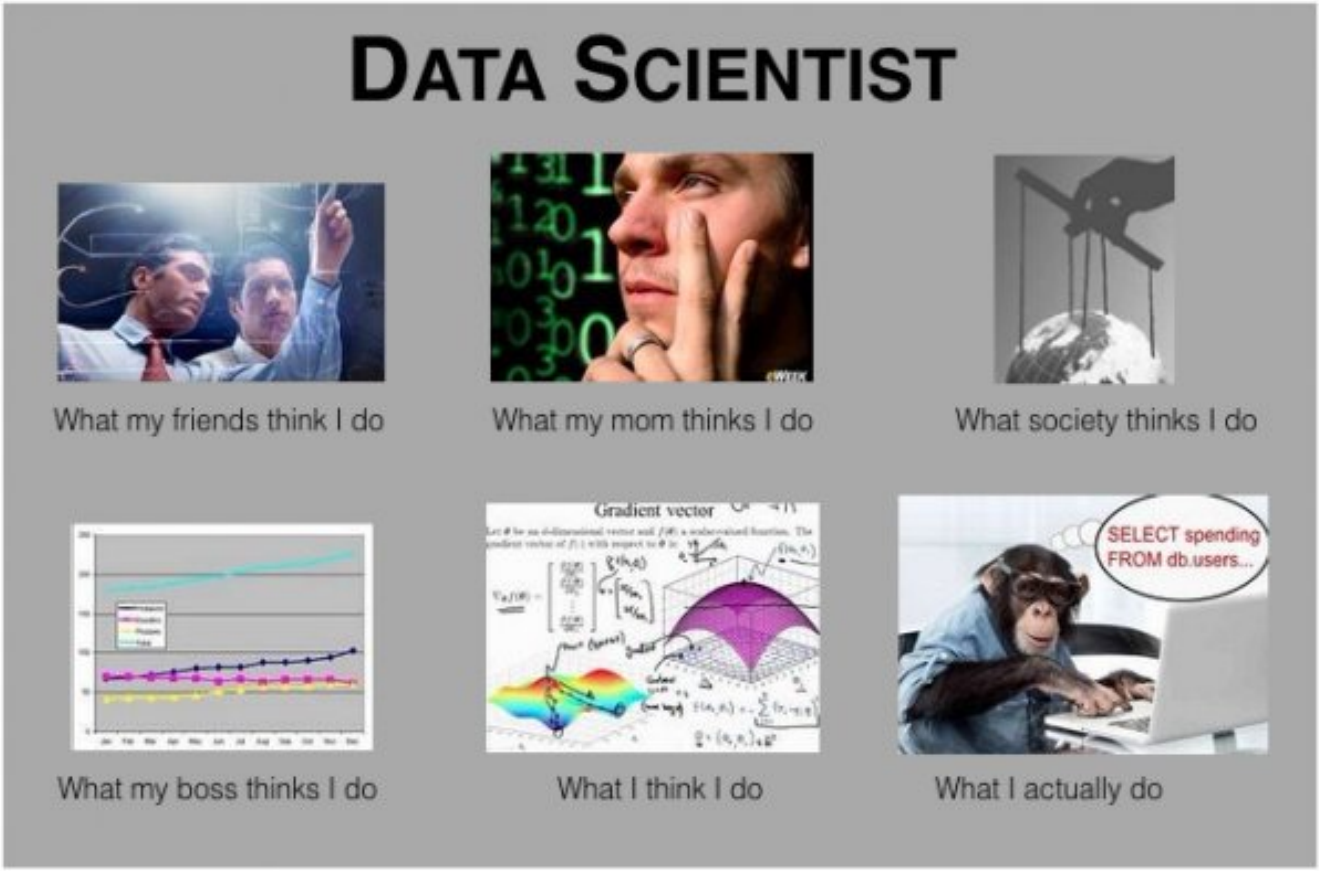
Projeto de Ciência de Dados — 2018

GRUPO #18



Hélio Domingos, 83473
Miguel Regouga, 83530
João Pina, 85080

ÍNDICE

1. INTRODUÇÃO

A área de Ciência de Dados envolve a computação de conjuntos de dados (*datasets*) e a descoberta e análise dos seus padrões, usando aprendizagem automática (*machine learning*) em harmonia com estatística.

O projeto apresentado aos alunos de Ciência de Dados consiste na aplicação de conhecimento sobre técnicas de *mining* e classificação de dados, de forma a descobrir informações e padrões sobre dois conjuntos de dados distintos.

O primeiro *dataset* em análise corresponde a dados recolhidos durante o uso diário de camiões da marca Scania, sendo o foco principal no sistema de pressão de ar (APS) dos mesmos, que tem como função gerar ar pressurizado a ser utilizado em várias funções do camião. A classe positiva do *dataset* consiste nas falhas dos componentes de um dado componente específico do sistema APS; já a classe negativa consiste em camiões com falhas em componentes não relacionadas com o sistema APS. O *dataset* tem, no total, 60.000 instâncias e 171 atributos.

O segundo *dataset* a analisar explora e estuda a avaliação subjetiva da qualidade de colposcopias digitais, registadas por médicos profissionais no Hospital Universitário de Caracas. O *dataset* está dividido em três modalidades, correspondentes às modalidades possíveis de colposcopias — Hinselmann, Green e Schiller. O número total de instâncias do *dataset* é de 287, e de atributos é de 69.

Em aprendizagem automática, existem dois métodos predominantes — **aprendizagem supervisionada** (*supervised learning*), que corresponde na análise de um conjunto de treino e, através de um dado algoritmo, processa uma função contingente que pode prever novas variáveis, dados novos exemplos; e **aprendizagem não supervisionada** (*unsupervised learning*), onde só é dado um conjunto de dados de entrada e nenhum conjunto de variáveis de saída, sendo que o objetivo é modelar a estrutura ou distribuição subjacente no conjunto de dados para aprender mais sobre eles — ambos irão ser abordados aquando do desenvolvimento do projeto.

Em relação à aprendizagem supervisionada (*classification*), irão ser estudados e comparados os seguintes algoritmos:

- **KNN (*Instance-Based Learning*)** — é o mais simples algoritmo de aprendizagem automática; "Baseia-se no princípio de que as amostras que são semelhantes, geralmente estão em proximidade (…) baseadas na aprendizagem por semelhança, ou seja, comparando uma amostra de teste com as amostras de treinamento disponíveis que lhe são semelhantes.
- **Naïve Bayes** — é geralmente usado para criar previsões baseadas em informações anteriores e apresentar evidências. Usando o conjunto de dados de treino, é possível adivinhar a probabilidade sem evidência adicional. O teorema de Naïve Bayes é: (P(B|A) x P(A)) / P(B), sendo A o conjunto de eventos de resultados categóricos e B a série de preditores.
- **Decision Trees —** é o algoritmo que permite o mapeamento de um conjunto de observações de um determinado conjunto de dados de treino. São técnicas de indução de mineração de dados que particionam recursivamente um conjunto de dados de registos usando uma abordagem *depth-first* gananciosa ou *breadth-first* até que todos os itens de dados pertençam a uma determinada classe.
- **Random Forests —** é um algoritmo que cria várias árvores de decisão e junta-as de forma a obter uma previsão mais precisa e estável.

Nos métodos de aprendizagem não supervisionada (*unsupervised mining*), irão ser estudados e comparados os seguintes algoritmos:

- **Association Rules —** é um método de aprendizagem automática baseado na utilização de regras, que permite descobrir relações interessantes entre variáveis em grandes conjuntos de dados, e destina-se a identificar regras fortes descobertas em *datasets* algumas medidas de interesse.
- **Clustering —**é o conjunto de técnicas de *data mining* que visa conceber agrupamentos automáticos de dados segundo o seu grau de semelhança; a cada conjunto de dados resultante do processo dá-se o nome de grupo, aglomerado ou agrupamento (*cluster*).

O sucesso de qualquer algoritmo depende sempre da qualidade do *input*. Assim, é necessário que os dados passem por um procedimento de pré-processamento, que permite, entre outras coisas, evitar a produção de elevados números de regras e a descoberta de padrões infrutuosos. Na fase de exploração, faremos uma análise dos resultados tendo em conta vários valores dos parâmetros de interesse, sendo para isso calculadas e comparadas diversas medidas estatísticas de interesse, tais como:

- ***Accuracy*****,** que corresponde à capacidade de o classificador categorizar corretamente todas as amostras. Fórmula: (TP + TN) / (TP + FP + FN + TN);
- ***Precision***, que corresponde à capacidade de não classificar como positiva uma amostra que é negativa. Fórmula: TP / (TP + TF);
- ***Sensibility*** (ou *Recall*), que corresponde à capacidade de o classificador encontrar todas as amostras positivas. Fórmula: TP / (TP + FN).

2. PRÉ-PROCESSAMENTO

2.1 Problema 1 — APS Failure

2.1.1 Unsupervised Mining

Aquando do pré-processamento do *dataset* relativo ao sistema APS dos camiões da Scania para a utilização de técnicas de *unsupervised mining*, foi-nos útil unificar ambos os *datasets* de teste e de treino num único, uma vez que com a utilização de técnicas de *unsupervised mining* não estamos a classificar, e, portanto, não é necessário haver dois *datasets* distintos. Assim, são-nos concedidos mais dados para analisar.

Apesar do atributo '*class'* ser só utilizado para classificação, no caso das *association rules* foi fundamental manter este atributo de forma a podermos realizar *feature selection*, dada a enorme quantidade de atributos. Já para o *clustering*, esta coluna foi descartada.

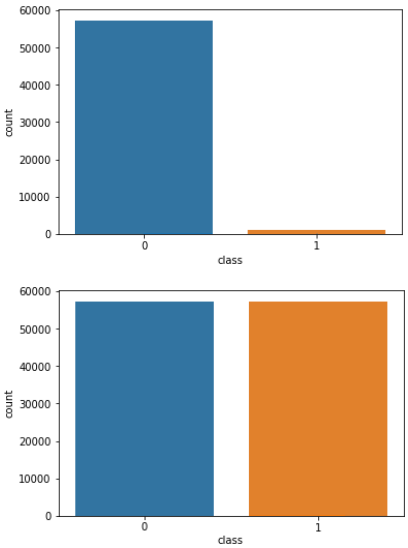
Ao analisar o *dataset*, descobrimos que existe uma grande quantidade dados que podem ser considerados *outliers*. *Outliers* são valores extremos que se desviam de outras observações num dado conjunto de dados, e podem indicar uma variabilidade numa medição, erros experimentais ou uma novidade. Em outras palavras, um *outlier* é uma observação que diverge de um padrão geral numa amostra, e a sua remoção é importante para que possamos encontrar padrões de dados mais concretos e precisos.

O dataset contém também um grande número de *missing values*, situação essa que é indesejável e tem de ser processada. Em relação a soluções, equacionámos a remoção desses dados omissos – contudo, esta decisão poderia ter um impacto negativo na performance dos métodos de aprendizagem.

Deste modo, em cada *missing value*, foi também analisada a solução de imputar a média dos valores presentes nessa coluna – o que, tendo em conta a anterior remoção dos *outliers*, é uma solução promissora e a considerar.

Para aplicar os algoritmos de clustering, fomos forçados a particionar o *dataset* em 10 amostras mais pequenas, sendo que para o fazer usámos *random sampling* sem reposição. Este procedimento foi executado de forma a evitar problemas de escassez de memória aquando da execução do algoritmo k-means.

2.1.2 Classification



Para a utilização de técnicas de classificação, foi aplicada uma transformação de discretização de dados ao *dataset*, correspondente à transformação do atributo binário "*class*", cujas insígnias "*pos*" e "*neg*" foram convertidos para numéricos 0 e 1, respetivamente.

Os *outliers* têm também de ser tratados com a utilização de técnicas de classificação. Analisámos por isso os *outliers* dos conjuntos de treino e de teste, separadamente, e tomamos o mesmo procedimento utilizado para o pré-processamento do *dataset* para a utilização em *unsupervised mining*. O mesmo aconteceu para os *missing values* — foi imputada a média dos valores.

Uma grande alteração que foi feita foi o balanceamento dos dados. Analisando o *dataset*, era possível verificar que o mesmo era preferencial, o que significa que tendia para a classe mais popular — no nosso caso, o '*neg'* (como representado no *Gráfico 1*). Foi, portanto, necessário reduzir as quantidades globais, não tendo em conta a distribuição de dados. A fim de equilibar os dados e evitar o *overfitting*, optámos por utilizar o SMOTE.

2.2 Problema 2 — Colposcopias

2.2.1 Unsupervised Mining

Para o processamento do segundo *dataset*, e dado que corresponde a um conjunto de dados bastante inferior ao primeiro, não foram necessários muitos procedimentos. No que toca ao pré-processamento para a utilização em técnicas de *unsupervised mining*, foram removidas as várias *features* '*expert'* e a *feature* '*consensus'*, uma vez que estas apenas nos são úteis na utilização de métodos de aprendizagem supervisionada (classificação).

Juntámos também os três *datasets* num só, uma vez que os três têm o mesmo conjunto de atributos e podem ser tratados como um só, uma vez que os parâmetros (atributos) são todos iguais, já que têm a ver com as características específicas de cada utente.

2.2.2 Classification

Para a utilização de técnicas de classificação, apenas foi necessário unificar os três *datasets* num único, para que sejam analisados de forma integrada, pela mesma razão apontada anteriormente.

3. EXPLORAÇÃO

3.1 Problema 1 — APS Failure

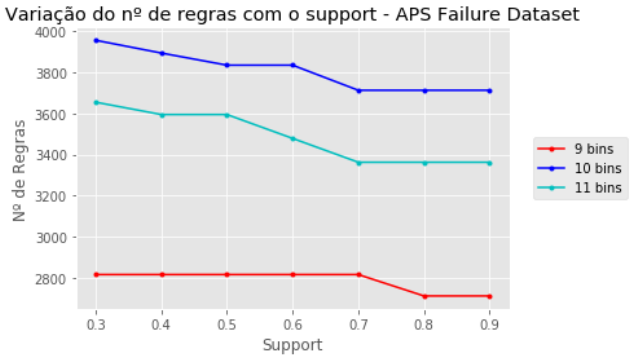
3.1.1 Unsupervised Mining

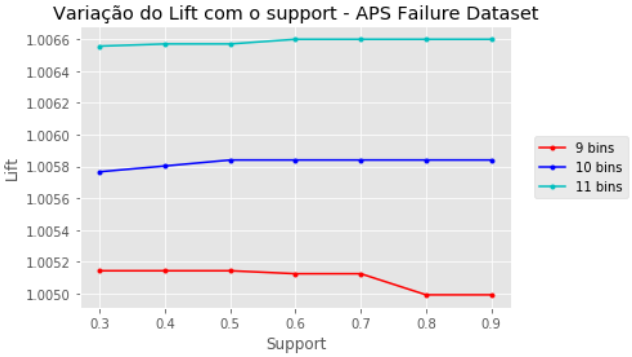
3.1.1.1 Association rules

Aquando da análise de métodos de *association rules*, começámos por realizar *feature-selection* com *decision tree* de modo a extrair as *features* mais relevantes das 170 do conjunto de dados, algumas das quais poderiam ser pouco pertinentes e poderiam influenciar negativamente a performance do algoritmo, gerando regras que não nos levam a qualquer conclusão.

Decidimos discretizar o conjunto de dados por 9, 10 e 11 *bins*, uma vez que a maioria dos atributos era do tipo *float*, o que significava que haveria uma quantidade baixa de conjuntos de itens frequentes quando tentássemos aplicar técnicas de *association rules*.

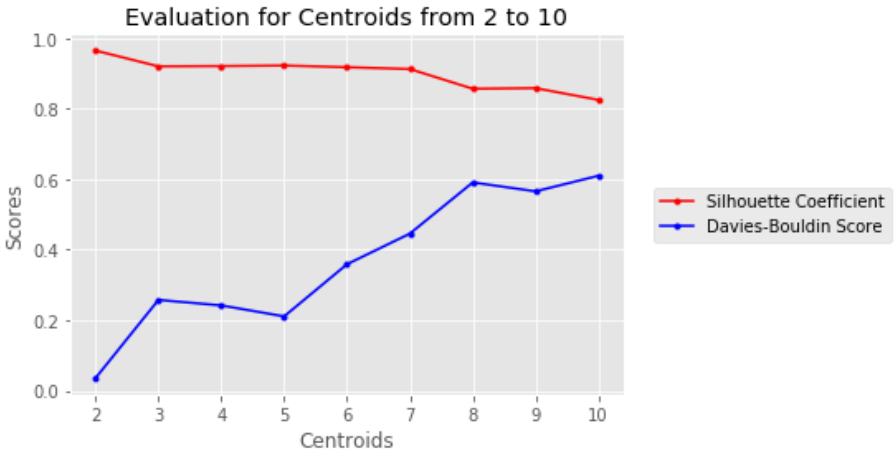
Definimos como 30% o valor mínimo para o suporte, e para a confiança mínima fixa o valor de 50%, sendo que a partir desses valores iterámos o suporte em 10%, até ao valor de 90%, aplicando a cada um destes suportes o algoritmo *apriori*. Tivemos em consideração apenas as regras cujo valor de *lift* seja maior que 1, uma vez que regras iguais ou inferiores a este valor não têm qualquer interesse (neste *dataset* não obtivemos regras com *lift* superior a 1.1).





3.1.1.2 Clustering

Para a utilização de métodos de *clustering* no primeiro *dataset*, usámos o algoritmo *k-means* sobre o *dataset* pré-processado, com um número de *centroids default* de 2, sendo que incrementámos iterativamente esse número até 10. Dado que estamos a analisar um *dataset* de grandes proporções, foi necessário particioná-lo em conjuntos de dados mais pequenos, de modo a evitar problemas de escassez de memória, como referido na secção de pré-processamento.



Comparámos a performance com e sem PCA, sendo que para este *dataset* os melhores resultados foram obtidos sem a utilização de PCA.

Olhando para o *Silhoutte Coefficient*, a média para as 10 amostras foi de 0.9643, e concluímos assim que os melhores resultados (valores mais altos) desta métrica foram obtidos para quando o número de *centroids* é igual a 2.

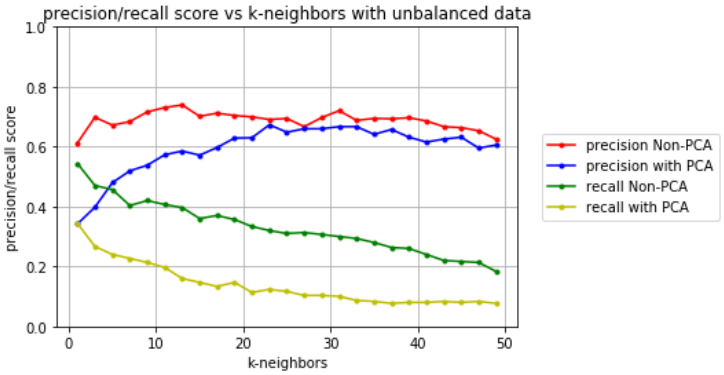
Analisando o *Davies-Bouldin Score*, a média para as 10 amostras foi de 0.0354, concluindo assim que os melhores resultados (valores mais baixos) desta métrica foram obtidos para quando o número de *centroids* é também igual a 2.

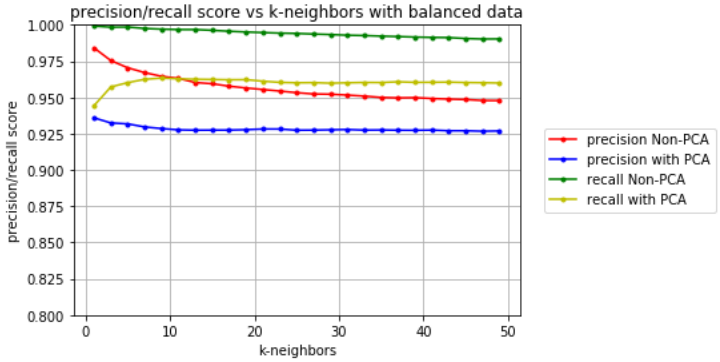
Para este *dataset*, concluímos que o número de *centroids* ideal é igual a 2.

3.1.2 Classification

3.1.2.1 Instance-based learning

Para a análise com o algoritmo KNN (*instance-based learning*), utilizámos os *datasets* com e sem PCA com variações do valor k-neighbors de 1 a 51, de 2 em 2, além de aplicar SMOTE para balancear os dados, tendo obtido os seguintes gráficos:





Podemos assim concluir que os melhores resultados de *precision* e *recall* são obtidos quando o conjunto de dados está balanceado com recurso ao SMOTE, quando não é utilizado PCA (vermelho e verde), e quando o *k-neighbors* é igual a 1. Com a utilização de PCA (azul e amarelo), os resultados das métricas tendem a piorar.

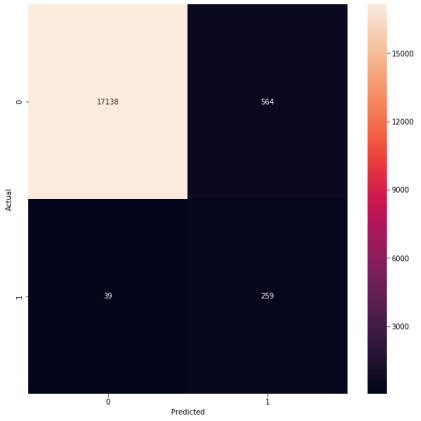
Analisando os resultados de *precision* e *recall*, podemos observar que a utilização de SMOTE melhorou significativamente o recall em relação ao dataset não-equilibrado, ou seja, o modelo tem uma melhor capacidade de identificar samples positivas. A precision é também ligeiramente melhor com SMOTE, o que significa o modelo tem uma boa capacidade de não classificar como positiva uma amostra que é negativa.

Com base na análise dos gráficos obtidas, podemos concluir que o desempenho do classificador KNN estabiliza a partir de k=20 e obtemos melhores resultados se for feito *oversampling* do conjunto com SMOTE, sendo que esse ponto estabilização do desempenho poderá estar relacionado com o grau de variedade média entre os atributos do *dataset*.

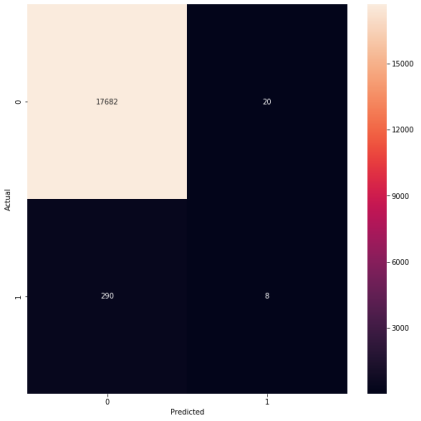
3.1.2.2 Naïve Bayes

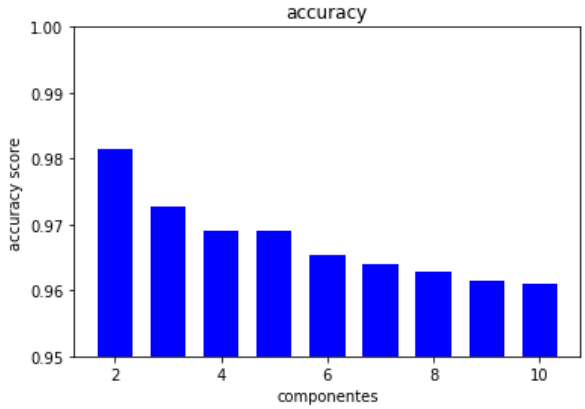
Relativamente ao Naïve Bayes, utilizámos o algoritmo que usa a distribuição gaussiana de probabilidades. Foi variado o número de componentes do PCA de forma a determinar qual o que produzia os melhores resultados. Para comparar os resultados, usámos as métricas de *accuracy* e *cross-validation*, tendo sido produzidos os seguintes gráficos:

Sem PCA



Com PCA



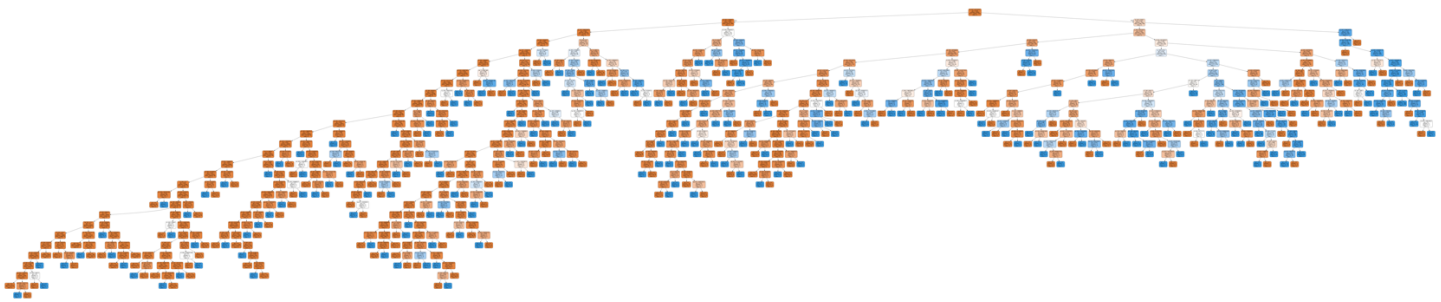


Da análise dos gráficos, podemos concluir que são obtidos melhores resultados quando é utilizado PCA com 3 componentes, sendo os melhores resultados alcançados com o *expert 4*.

3.1.2.3 Decision Trees

Para a construção das *Decision Trees* no primeiro *dataset*, foi utilizado o algoritmo CART (*Classification and Regression Trees*), que usa o Gini como métrica. Para testar fizemos *pre-pruning* (de modo evitar o *overfitting*, isto é, tentar parar o processo de construção de árvores precocemente antes de produzir folhas com amostras muito pequenas) variando dois dos atributos do classificador: *min_samples_split* e *min_samples_leaf*. O *min_samples_leaf* é o número mínimo de amostras que uma "folha" da árvore tem de ter para existir. O *min_samples_split* é o número mínimo de amostras que um "nó" tem de ter para se subdividir. Após uma investigação e uma variação entre 1 (no caso do *min_samples_leaf*) ou 2 (no caso do *min_samples_split*) e 3000 dos dois atributos, concluímos que o número ideal para o *min_samples_leaf* é de 1 e para o *min_samples_split* é de 2.

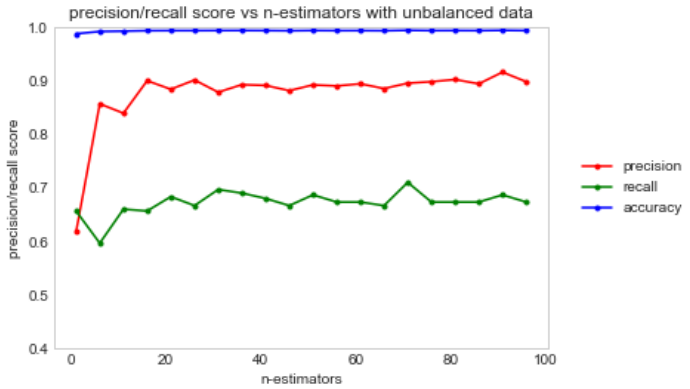
Aquando da construção das árvores de decisão, utilizámos *feature selection* de modo a filtrar quais as *features* que são mais úteis para o algoritmo executar a classificação. De facto, ao analisar, descobrimos features que tinham nula ou pouca importância (abaixo de 0.01%). Usámos, portanto, o modelo *select from model* para determinar as *features* mais importantes, sendo essas as que foram realmente utilizadas.

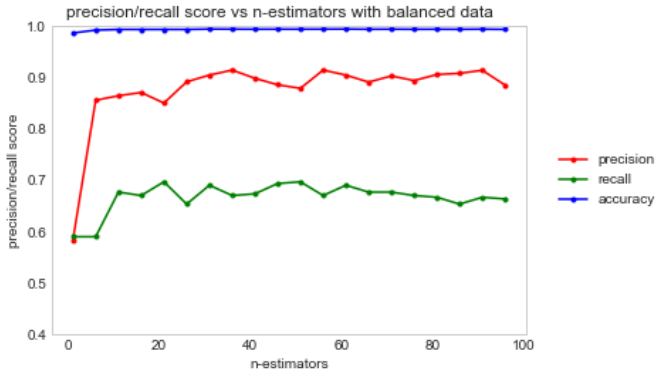


3.1.2.4 Random Forests

O método *Random Forests* consiste num método de aprendizagem de conjunto usado para classificação, cuja operação se resume a um meta-estimador que se ajusta a vários classificadores de árvore de decisão em várias subamostras do conjunto de dados e usa a média para melhorar a precisão preditiva e controlar o ajuste excessivo.

Analisámos a performance do algoritmo com o nosso *dataset* ao variar o número de "árvores na floresta" (*n_estimators*) entre 1 e 100, de 5 em 5, com o *dataset* balanceado e não balanceado:





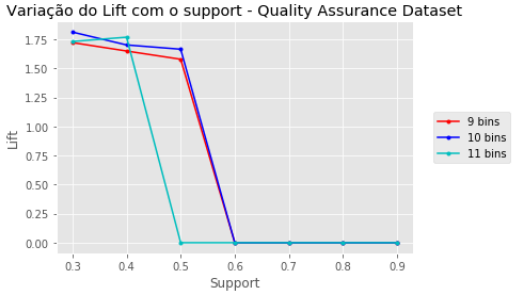
Podemos verificar que em ambos os casos a *accuracy* está muito perto dos 99%. Isto pode dever-se à elevada quantidade de amostras do *dataset* — esta característica permite que o classificador tenha muitos dados para aprender, o que o torna capaz de classificar corretamente a maior parte das amostras.

3.2 Problema 2 — Colposcopias

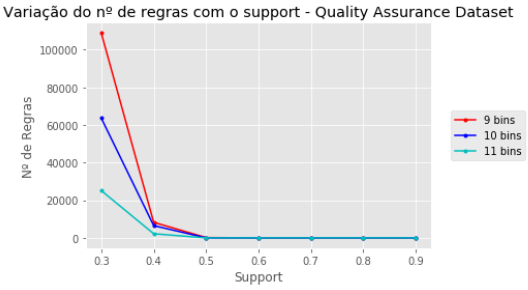
3.2.1 Unsupervised Mining

3.2.1.1 Association rules

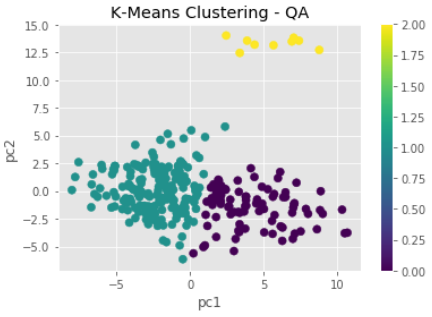
Aquando da análise de métodos de *association rules* no segundo *dataset*, discretizámos o conjunto de dados por 9, 10 e 11 *bins*, uma vez que todos os atributos são do tipo *float*, o que significa que há uma quantidade baixa de conjuntos de itens frequentes quando tentássemos aplicar técnicas de *association rules*.

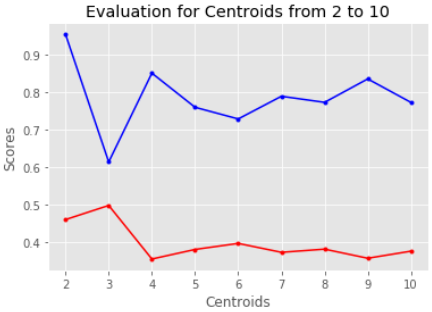


Definimos como 30% o valor mínimo para o suporte, e para a confiança mínima fixa o valor de 50%, sendo que a partir desses valores iterámos o suporte em 10%, até ao valor de 90%, aplicando a cada um destes suportes o algoritmo *apriori*. Tivemos em consideração apenas as regras cujo valor de *lift* seja no mínimo 1.5, uma vez neste *dataset* temos bastantes regras cujo lift é superior a 1, o que nos permite obter regras mais pertinentes.



3.2.1.2 Clustering





Para a utilização de métodos de *clustering* no segundo *dataset*, usámos também o algoritmo *k-means* sobre o *dataset* pré-processado, com um número de *centroids default* de 2, sendo que incrementámos iterativamente esse número até 10.

Comparámos a performance com e sem PCA, sendo que para este *dataset* os melhores resultados foram obtidos **com** a utilização de PCA, com o número de componentes igual a 2.

Olhando para o *Silhoutte Coefficient*, a média para as 10 amostras foi de 0.4980, e concluímos assim que os melhores resultados (valores mais altos) desta métrica foram obtidos para quando o número de *centroids* é igual a 3.

Analisando o *Davies-Bouldin Score*, a média para as 10 amostras foi de 0.6137, concluindo assim que os melhores resultados (valores mais baixos) desta métrica foram obtidos para quando o número de *centroids* é também igual a 3.

Para este *dataset*, concluímos que o número de *centroids* ideal é igual a 3.

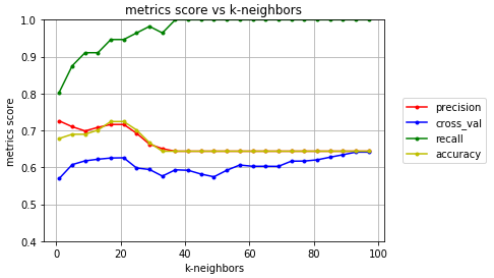
3.2.2 Classification

3.2.2.1 Instance-based learning

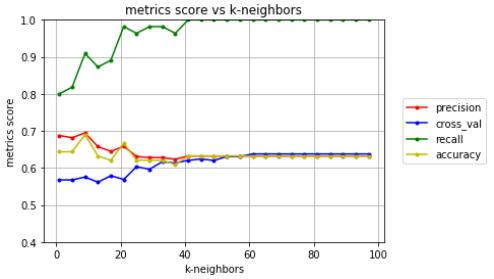
Em relação ao *dataset* das colposcopias, existem 7 colunas de classificação — 6 que representam a opinião subjetiva de profissionais da área e 1 que corresponde à maioria das opiniões. Dado que são juízos subjetivos, poderão haver opiniões "mais certas do que outras", isto é, poderão haver profissionais que se enganam na sua opinião.

Tendo isto em conta, analisámos a performance do algoritmo KNN (instance-based learning) com cada uma das 7 colunas, de modo a verificar a capacidade de avaliar corretamente o problema, tendo sido obtidos os seguintes gráficos:

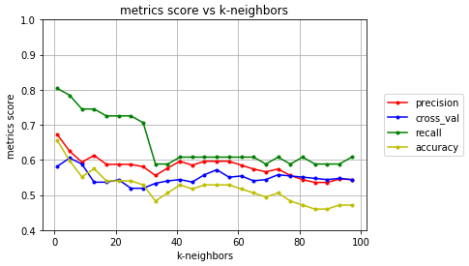
Experts 0



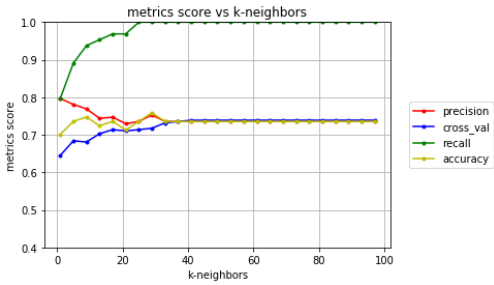
Experts 1



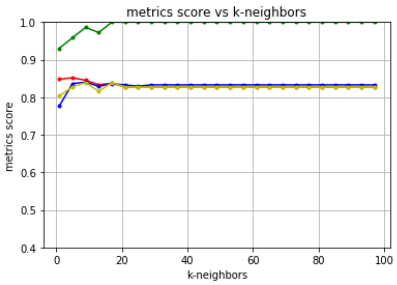
Experts 2



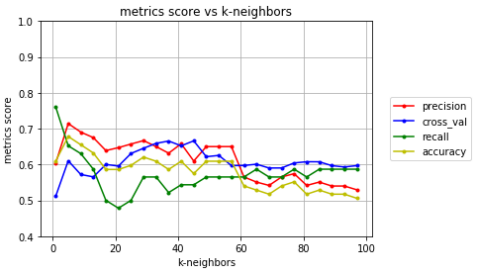
Experts 3



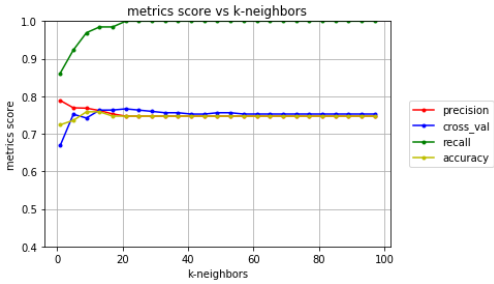
Experts 4



Experts 5



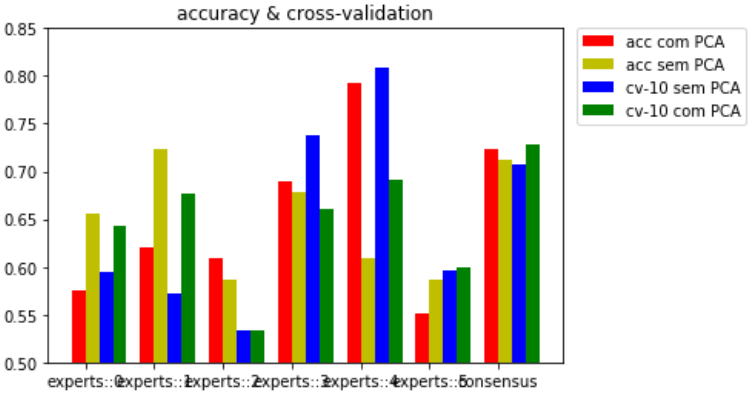
Consensus

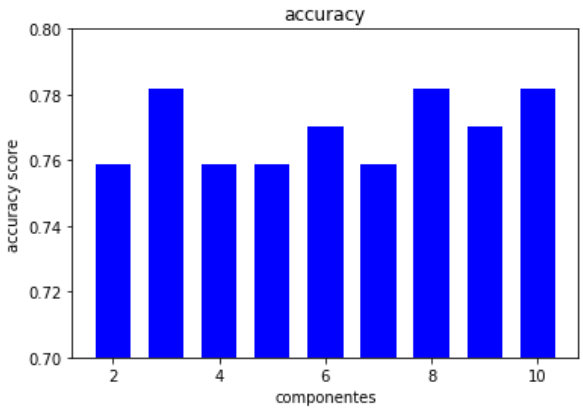


Da análise dos gráficos correspondentes às 6 colunas, podemos concluir que o Expert 4 é o que tem uma melhor capacidade de classificar corretamente todas as amostras, ou seja, é o que tem uma *accuracy* mais elevada, sendo o mais "correto" de entre os outros experts. É de notar que o Expert 4 consegue obter resultados muito melhores do que o *consensus*, sendo que a performance desta *feature* poderá ser melhorada se for atribuído um menor peso às colunas de experts com piores resultados (como as dos experts 1, 2 e 5).

3.2.2.2 Naïve Bayes

Relativamente à aplicação de Naïve Bayes no segundo *dataset*, utilizámos também o algoritmo que usa a distribuição gaussiana de probabilidades. Foi variado o número de componentes do PCA de forma a determinar qual o que produzia os melhores resultados. Para comparar os resultados, usámos as métricas de *accuracy* e *cross-validation*, tendo sido produzidos os seguintes gráficos:

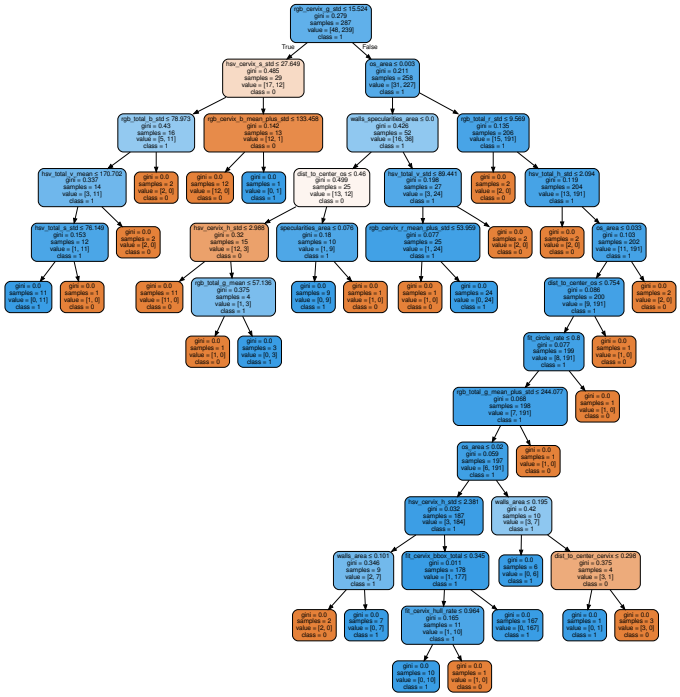




Da análise dos gráficos, podemos concluir que são obtidos melhores resultados quando é utilizado PCA com 2 componentes.

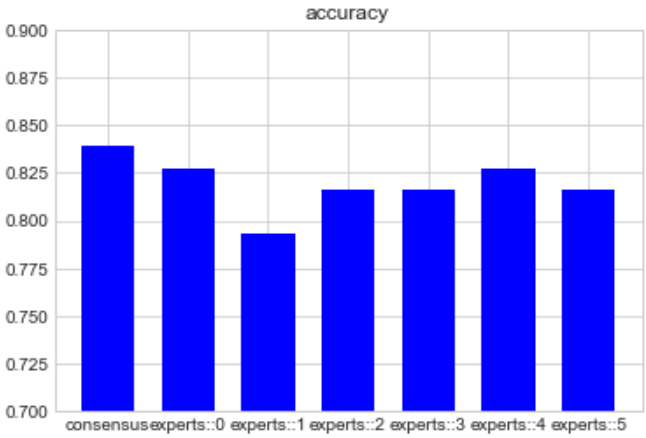
3.2.2.3 Decision Trees

Para a construção das *Decision Trees* no segundo *dataset*, foi também utilizado o algoritmo CART (*Classification and Regression Trees*), que usa o Gini como métrica. Para testar fizemos *pre-pruning* variando dois dos atributos do classificador: *min_samples_split* (sendo o número ideal igual a 2) e *min_samples_leaf* (com número ideal igual a 1). Tal como no primeiro *dataset*, usámos *feature selection* de modo a filtrar quais as features que são mais úteis para o algoritmo executar a classificação.



Ao gerar a árvore para o expert 4 (representada à direita), reparámos que a métrica Gini é mais alta quando FL <= 15.524, sendo por isso esse nó escolhido como raiz da árvore. Olhando para o segundo nível da árvore, temos diferentes valores de Gini para um mesmo atributo, tendo um valor mais alto num intervalo mais circunscrito de CW — isto acontece porque o Gini é calculado para cada caso específico. Em cada decisão, a seta à esquerda corresponde a *True* e à direita corresponde a *False*, tal como é verificado na raiz da árvore.

3.2.2.4 Random Forests



Para o segundo *dataset*, analisámos a performance do algoritmo Random Forests ao variar o número de "árvores na floresta" (n_*estimators*) entre 1 e 100, de 5 em 5, com o *dataset* balanceado e não balanceado. A comparação entre as diferentes *target variables* encontra-se representada no gráfico à esquerda.

Foi obtida uma prestação melhor quando o *n_estimators* foi igual a 70, sendo que podemos verificar que a métrica de *accuracy* é melhor quando a *target variable* é a coluna *consenus*.

Estes resultados são interessantes quando comparados, por exemplo, com o classificador KNN, em que o *consensus* tem uma *accuracy* ligeiramente inferior ao melhor expert (em ambos os casos, o expert 4).

4. ANÁLISE CRÍTICA

Acerca do *dataset* correspondente às colposcopias, foi considerado analisar separadamente cada ficheiro, ao invés de como um todo. Esta opção foi descartada, uma vez que, apesar de serem métodos de colposcopia diferentes, os parâmetros são idênticos e os valores medidos também, sendo que considerámos que a análise separada não traria qualquer vantagem e/ou diferença nos resultados. Contudo, posteriormente, verificámos que, de facto, ao analisar separadamente o dataset, poderiam ser obtidos melhores resultados de accuracy. Além disso, os experts que se enganam mais variam de técnica para técnica, algo que deveríamos também ter tido em conta.

Ainda sobre o pré-processamento, considerámos remover os atributos que continham mais de 65% de *missing values* e com mais de 90% de zeros. Contudo, foi decidido não adotar esta medida, uma vez que a remoção destes valores pode ter um impacto indesejável aquando da utilização das diversas técnicas. Foi por isso escolhido imputar a média de cada atributo nos *missing values*, que, apesar de ter as suas desvantagens (uma vez que pode reduzir a variabilidade dos dados) foi a que considerámos fazer mais sentido no nosso caso. Poderíamos, no entanto, ter explorado e analisado outras técnicas de tratamento de *missing values*, como Z-score, modelos de regressão linear, entre outros.

O conjunto de dados correspondente às falhas do sistema APS é colossal, o que levou a que algumas medidas fossem tomadas — entre elas, a divisão em samples, para evitar diversos problemas de falta de memória. Além disso, o dataset era bastante desequilibrado — neste aspeto, o SMOTE foi crucial para que os resultados obtidos fossem consistentes e precisos. Contudo, o SMOTE não é perfeito, uma vez que gera samples com base nos existentes, que podem não ser representativos e podem também ser repetidos, levando a um enviesamento dos dados, o que leva a que a métrica de *accuracy* seja tão alta.

O algoritmo Naïve-Bayes assume que os atributos a considerar são independentes, e produz a classificação com base nesta assunção. Isto pode não ser verdade na grande maioria dos casos, uma vez que poderão existir atributos que são dependentes, o que levou a uma diminuição na qualidade dos resultados obtidos. Para os melhorar, foi utilizado PCA, que levou à melhoria da distribuição de probabilidades, e, consequentemente, a melhores resultados.

Em relação ao *clustering* no primeiro *dataset*, para as duas métricas utilizadas obtivemos um número ideal de *centroids* igual a 2, o que faz sentido analisando os dados, já que originalmente tínhamos duas classes que dividia o nosso conjunto de dados. Já em relação ao segundo *dataset*, obtivemos um número ideal de *centroids* igual a 3; no Naïve-Bayes, o número de componentes ideais no PCA também foi igual a 3, o que nos leva a concluir que cada técnica de colposcopia pode analisar cada atributo de uma forma diferente, e por esta razão poderá fazer sentido agrupar os dados em 3 formas distintas. Esta convicção é suportada pela análise com o Naïve-Bayes, em que encontrámos diferenças da análise de cada *expert* segundo as diferentes técnicas, isto é, cada *expert* tem um valor de *accuracy* diferente se forem analisadas as técnicas em separado ou em conjunto.

5. CONCLUSÕES

Vunc sed eede. Praesent vitae lectus. Praesent neque justo, vehicula eget, interdum id, facilisis et, nibh adsumes. Phasellus at purus et libero lacinia dictum. Fusce aliquet. Nulla eu ante placerat leo semper dictum.

REFERÊNCIAS

[1] Um breve resumo de técnicas de deteção de outliers, Towards Data Science — https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561
[2] Como detetar e remover *outliers*, Towards Data Science — https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba
[3] Como lidar com dados desaparecidos, Towards Data Science — https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4
[4] Métricas de Ciência de Dados, Towards Data Science — https://towardsdatascience.com/data-science-performance-metrics-for-everyone-4d68f4859eef
[5] Tutorial de PCA em Ciência de Dados, Dezyre — https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial
[6] Árvores de Decisão em Aprendizagem Automática, Towards Data Science — https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052
[7] Guia de Início Rápido para Árvores de Decisão, Towards Data Science — https://towardsdatascience.com/a-beginners-guide-to-decision-tree-classification-6d3209353ea
[8] Algoritmo Random Forest em Python, Towards Data Science — https://towardsdatascience.com/random-forest-in-python-24d0893d51c0