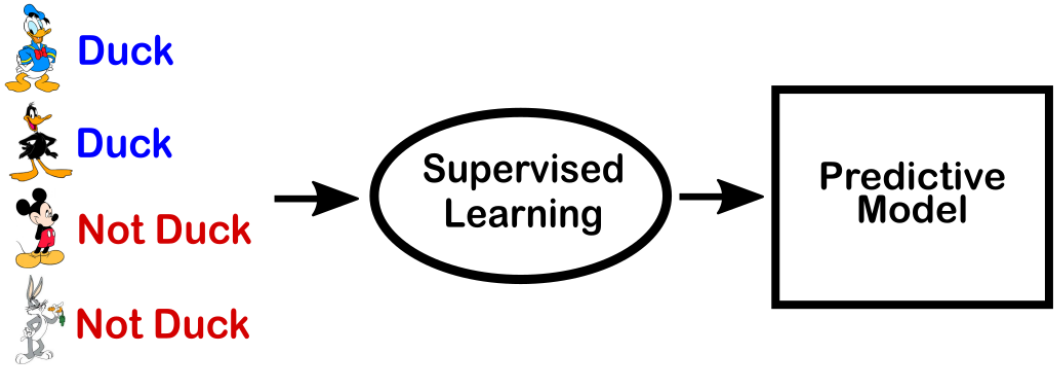
TÉCNICO LISBOA

Data Science - Project 2



Grupo	**46**
Daniel Correia	80967
Pedro Orvalho	81151
Renato Cardoso	81530

INDEX

1 PRE PROCESSING

1.1 Problem 1 - Crabs dataset

Neste dataset, aplicámos 2 transformações de normalização: converter o atributo binário "sex" de "M" e "F" para numéricos 0 e 1; converter o atributo binário "sp" de "B" e "O" para numéricos 0 e 1.

Também removemos a coluna "index" devido a esta feature ter sido gerada inicialmente a partir da espécie e do sexo de cada crab. Ou seja, esta feature é pouco útil para tentar adivinhar a espécie de um crab através dos processos de classificação.

Utilizámos ainda PCA de 2 componentes para gerar um dataset com menor dimensionalidade (também usámos PCA no no-shows).

Não utilizámos discretização por bins porque não era uma técnica apropriada para os classificadores utilizados: para o KNN seria impossível porque o algoritmo obriga a utilização de uma matriz de valores numéricos; para as Decision Trees não faz sentido porque este algoritmo já faz uma divisão por intervalos.

1.2 Problem 2 - No-shows dataset

Neste dataset, aplicámos 4 transformações de normalização: converter o atributo binário "Gender" de "M" e "F" para numéricos 0 e 1; converter o atributo binário "No-show" de "No" e "Yes" para numéricos 0 e 1 e alteŕamos o nome do atributo para "class"; converter as datas "AppointmentDay" e "ScheduledDay" para a semana do ano correspondente (1-52).

Também removemos as colunas "PatientId", "AppointmentID", "Neighbourhood" porque, no caso dos IDs, são atributos gerados automaticamente sem qualquer relação com as características do paciente. No caso do "Neighbourhood", não existe uma técnica de normalização/discretização que tenha em conta o contexto geogŕafico associado a este atributo, e visto que não estamos no Brasil nem sabemos a distância geográfica entre as localizações optámos por retirar esta coluna.

Para além disto, como este dataset está desequilibrado como uma distribuição do atributo "No-show" é 80% de "No" e 20% de "Yes", portanto aplicámos o algoritmo de over-sampling SMOTE sobre a porção de treino para tentar melhorar os modelos obtidos. Como Under Sampling utilizamos um algoritmo de random under sampler, que escolhe aleatoriamente o número de dados da classe com mais elementos, até ficar com o mesmo número de elementos que a classe menor.

2 EXPLORATION

2.1 Metrics fórmulas

Dado que utilizámos a biblioteca sklearn de Python para explorar os processos de classificação nos datasets, apresentamos de seguida as fórmulas das métricas utilizadas para clarificar o significado de cada métrica relativamente às métricas estudadas nas aulas teóricas.

Accuracy:

- Fórmula: (TP + TN) / (TP + FP + FN + TN)
- Significado: capacidade do classificador classificar corretamente todas as samples.
- Correspondência com a teórica: Accuracy.

Precision:

- Fórmula: TP / (TP + FP)
- Significado: capacidade de não classificar como positiva uma sample que é negativa.
- Correspondência com a teórica: não existe correspondência direta.

Sensibility (or Recall):

- Fórmula: TP / (TP + FN)
- Significado: capacidade do classificador encontrar todas as samples positivas.
- Correspondência com a teórica: Sensibility.

AUC ROC:

- "Area under the receiver operating characteristic curve"
- Significado: valor da área associada à ROC curve. permite avaliar o par sensitivity/specificity
- Correspondência com a teórica: não existe correspondência direta.

Decidimos não calcular os valores da métrica "Specificity" (por exemplo, a partir das confusion matrizes obtidas) porque não acrescentava muito valor à análise dos 2 datasets saber a capacidade de encontrar as samples negativas. No caso do no-shows, isto é óbvio porque o dataset tem uma maioria de samples negativas, estando o desafio na capacidade de escolher um classificador que consiga identificar as samples positivas que estão em minoria.

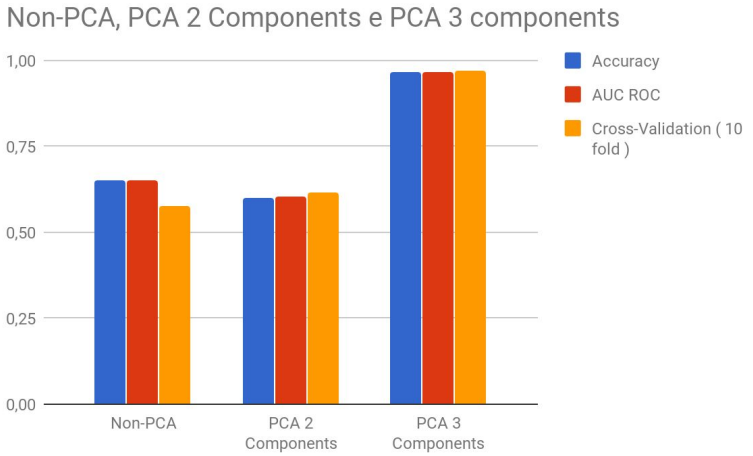
2.2 Problem 1 - Crabs dataset

Para todos os classificadores, escolhemos uma divisão de training/test de 70-30 e utilizámos 10-folds cross validation devido à dimensão reduzida do dataset (na ordem das centenas).

2.2.1 Naive-Bayes

Relativamente ao Naive-Bayes, a biblioteca utilizada continha 3 algoritmos para a realização do Naive-Bayes, cada um tratando os dados e as probabilidades de forma diferente. Sendo que foi escolhido o algoritmo que utiliza distribuição gaussiana de probabilidades porque foi leccionado nas aulas teóricas.

Variou-se ainda o número de componentes do PCA de forma a escolher aquele que produzia melhores resultados. Utilizámos como métricas de avaliação a accuracy, o AUC ROC e cross-validation. São agora apresentados os resultados obtidos:



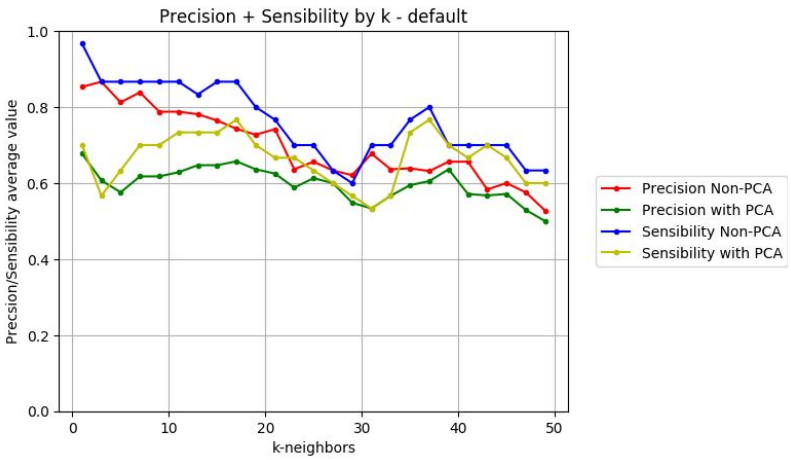
Como podemos ver nos gráficos, os resultados obtidos sem PCA e com PCA para 2 componentes são bastante baixos: cerca de 65% sem PCA e 60% com PCA.

Porém, para PCA com 3 componentes existe uma grande subida, porque com 2 componentes era considerada em maioria o sexo das espécies para a realização do PCA, o que conduzia a percentagem mais próxima dos 50-60%.

A partir de 3 componentes os valores de accuracy do AUC ROC e do cross-validation melhoram bastante, pois a distribuição dos dados é mais homogénea.

2.2.2 KNN

Para o KNN, utilizámos os datasets sem e com PCA com variações do valor de k-neighbors de 1 a 50, de 2 em 2.
De seguida, apresentamos os gráficos das métricas e discussão sobre estes.

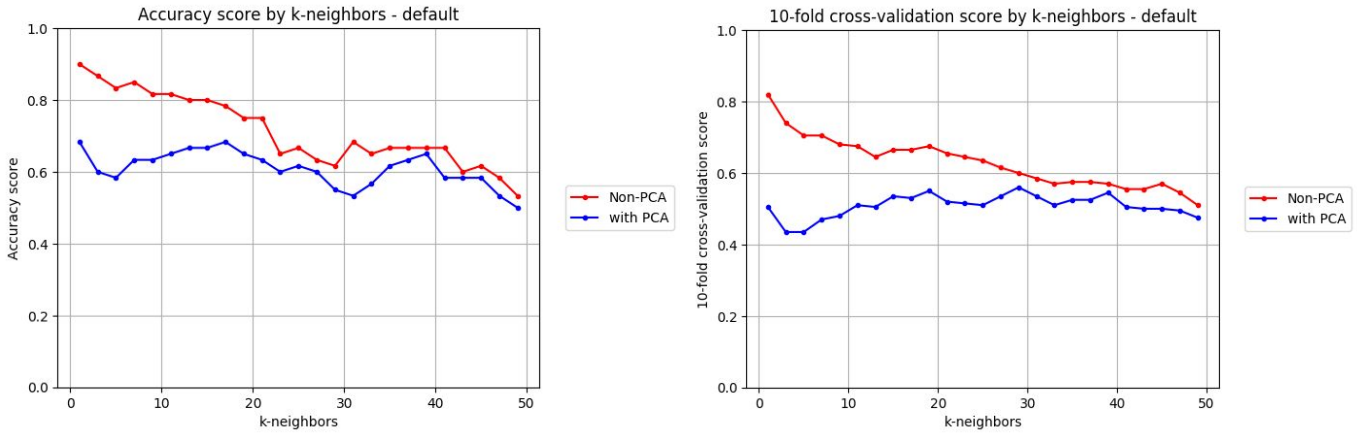


Podemos verificar que sem PCA os melhores resultados de precision e recall (vermelho e azul) são para k=1.

Com PCA (verde e amarelo) os resultados das métricas pioram, embora apresentem uma evolução semelhante.

Para além disso, analisando os resultados de accuracy e cross-validation, podemos tirar conclusões semelhantes: obtemos os melhores resultados com k=1; com PCA os resultados são piores do que sem PCA; a evolução de ambas as métricas decresce à medida que o k aumenta.

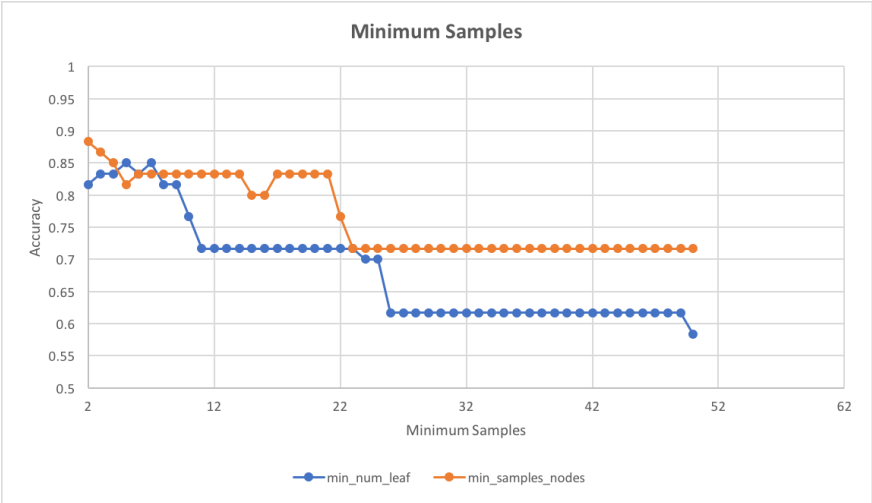
A diferença entre os valores destas métricas para k=1 é pequena (90% de accuracy vs ~81% de cross-validation) portanto k=1 parece ser uma boa escolha.



A ocorrência do k=1 como melhor resultado de classificação pode estar a ser causado pelo facto de a maior parte das caraterísticas dos crabs serem floats com uma variação muito baixa entre si (próxima de 1). Por outro lado, o tamanho do dataset é muito reduzido o que pode estar a induzir em erro a nossa análise devido à falta de amostras.

2.2.3 Decision Trees

Relativamente às Decision Trees, utilizámos o algoritmo CART que é disponibilizado em Python. Para testar fizemos pre-pruning variando dois dos atributos do classificador, min_samples_leaf e min_samples_split. 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.



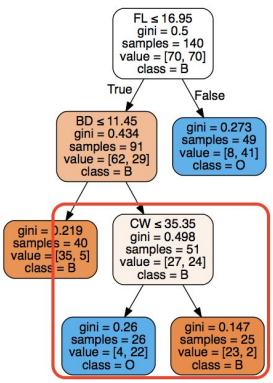
Podemos observar que as melhores accuracies são 85% quando o número de amostras mínimas nas folhas é 4, ou 89% quando nos nós é 2.

A accuracy vai aumentando e diminuindo conforme vamos perdendo níveis da árvore e consequentemente perdendo informação.

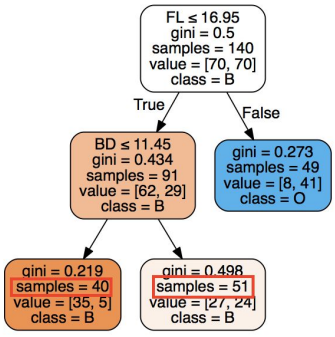
A accuracy diminui e depois mantém-se constante ao passar o número mínimo de amostras para folhas de 26 (linha azul).

Isto porque as árvores obtidas a partir do momento em que restringimos o número mínimo de amostras nos nós de 25 passam a variar os intervalos da mesma classe "B", perdendo o teste CW (primeira árvore). No caso das árvores min_sample_leaf=40 e 41, no teste ao BD são alterados os intervalos de valores das folhas de modo a mover amostras de uma folha para a outra, mantendo a mesma accuracy. A segunda folha perde três amostras (51 -> 48) de modo à primeira folha subir 3 amostras (40->43) e não se ter de diminuir a profundidade da árvore não perdendo informação e deste modo mantendo a accuracy.

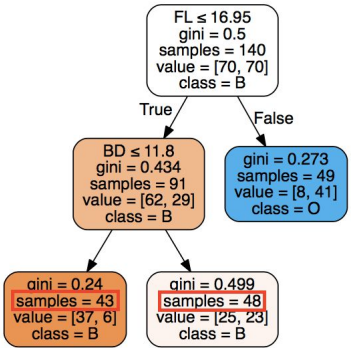
(min_sample_leaf = 25)



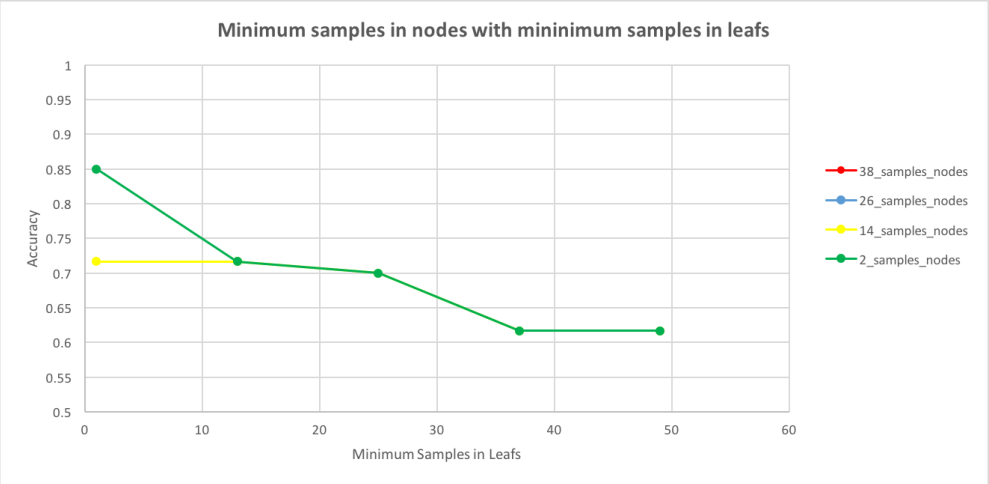
(min_sample_leaf = 40)



(min_sample_leaf = 41)

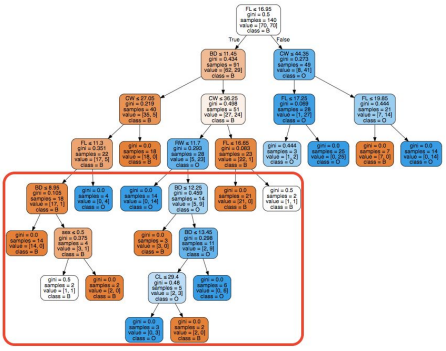


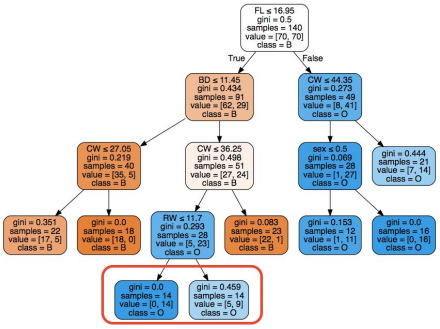
Testámos também fazer uma combinação de dois factores, aumentar o número de amostras mínimas necessárias para uma determinada folha existir para cada número de amostras mínimas nos nós, ou seja conforme a árvore vai diminuindo quer em termos de folhas quer em termos de nós.



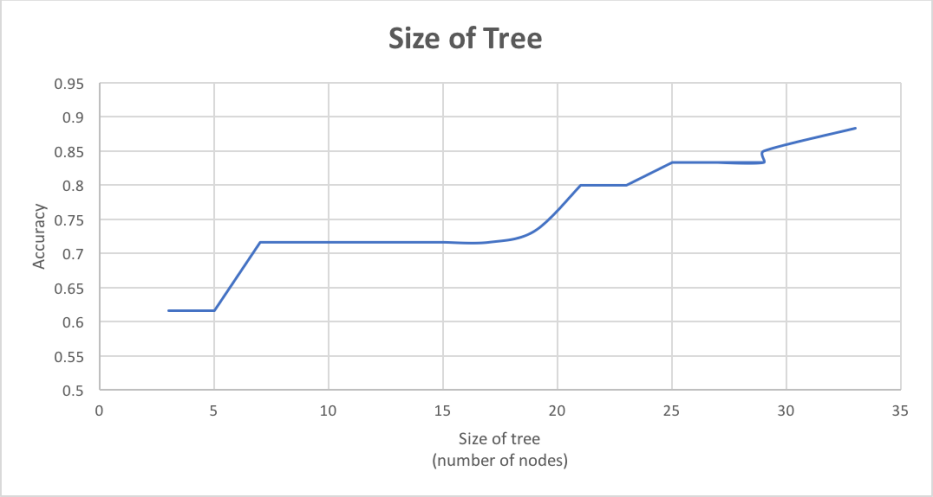
Obtivemos o gráfico à esquerda, onde houve uma diminuição de accuracy quando o min_samples_leaf passa de 2 para 12, isto porque como podemos observar pelas árvores em baixo, ao cortar folhas com menos de 12 amostras poda demasiado a árvore dado que o dataset é muito pequeno (200 amostras).

Logo cortar 12 amostras mínimas nas folhas provoca um decréscimo de quase 15% na accuracy.





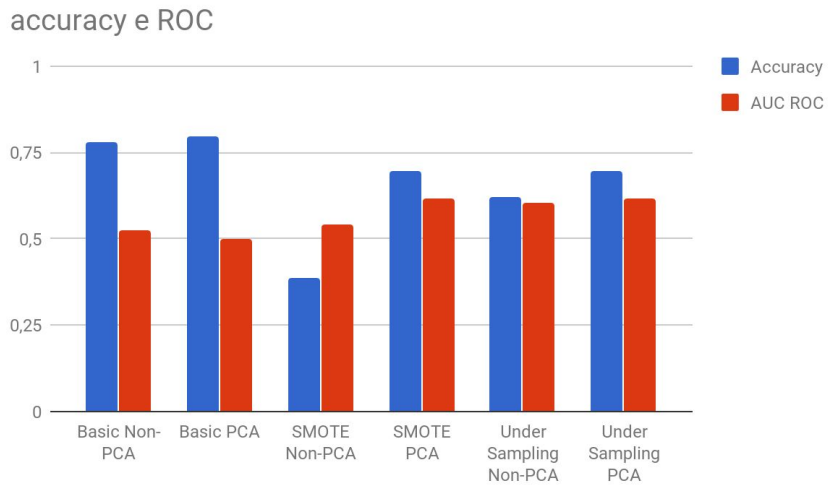
Por último, combinámos todos os testes anteriores para calcular a accuracy dependendo do tamanho da árvore e confirmámos que quanto maior for a árvore maior será a accuracy, porque a quantidade de informação contida na árvore aumenta, isto é, são feitos mais testes com os diferentes atributos, originando melhores resultados.



2.3 Problem 2 - No-shows dataset

2.3.1 Naive-Bayes

Relativamente ao Naive-Bayes, usamos o mesmo algoritmo, porém, apenas foram considerados a accuracy e o AUC ROC como métricas de avaliação (não se usámos cross-validation porque o dataset é muito grande).



Olhando para os resultados, é importante notar que em qualquer um dos casos a utilização de PCA melhorou os resultados, tanto da accuracy como do ROC.

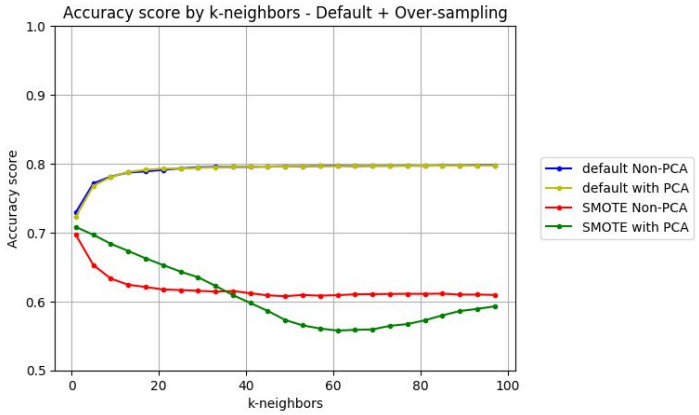
No primeiro caso (Basic PCA/Non-PCA), os resultados da accuracy foram tão altos porque, como não se aplicou nenhum método de oversampling ou undersampling, o algoritmo decidiu apostar naquele que tinha maiores chances de aparecer. Resultando numa grande discrepância entre a accuracy e o AUC ROC.

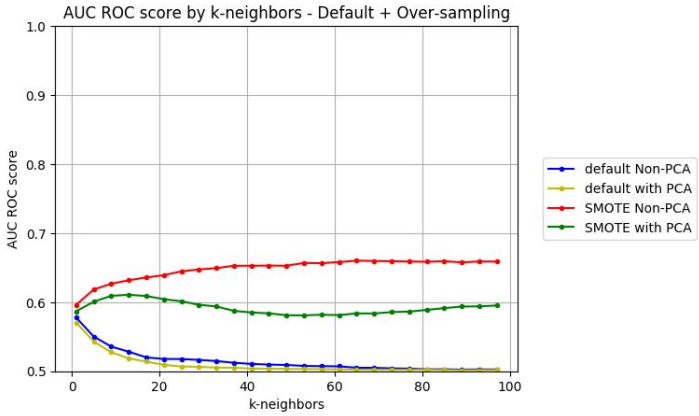
Quanto ao balanceamento, tanto o SMOTE como o método de undersampling, deram resultados parecidos com o respectivo PCA.

Sem PCA existe uma discrepância entre o SMOTE e o método de undersampling que pode ter acontecido porque com undersampling são eliminados dados redundantes, o que resulta num melhor valor de accuracy e de AUC ROC. Por outro lado, com o PCA, havendo um pré-processamento dos dados, esta discrepância é reduzida.

2.3.2 KNN

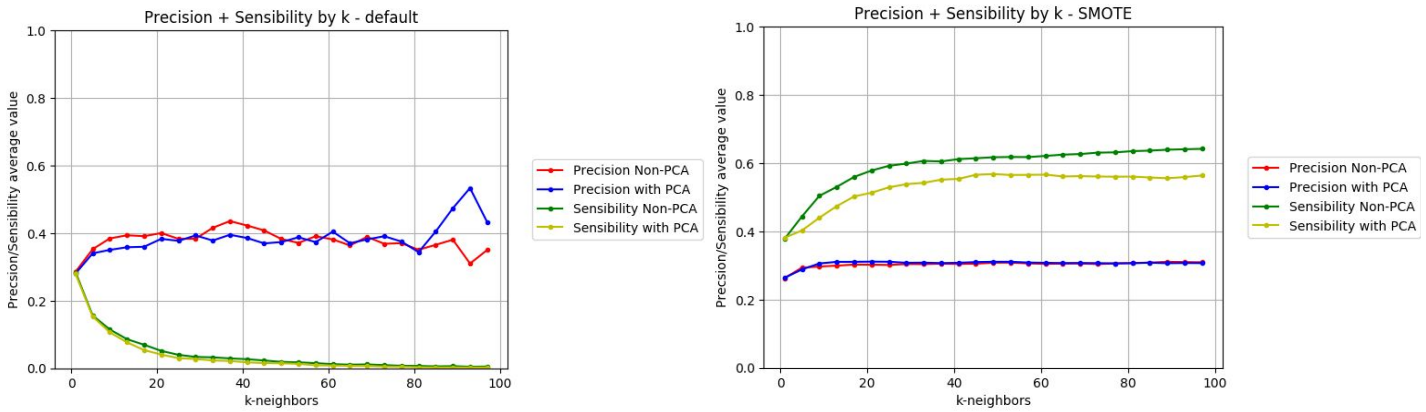
Para o KNN, utilizámos os datasets sem e com PCA com variações do valor de k-neighbors de 1 a 100, de 4 em 4. Também aplicámos SMOTE para equilibrar o dataset por over-sampling. De seguida, apresentamos os gráficos das métricas obtidas.





Como podemos observar (com escala [0.5,1]), a utilização de SMOTE piora os resultados a nível da accuracy em relação ao dataset não-equilibrado. Isto é normal porque o modelo é biased e dá preferência à escolha de negativos por pertencerem à maioria. No entanto, a AUC ROC tem uma relação inversa, ou seja, usar SMOTE melhora os resultados comparativamente aos do dataset não-equilibrado. Isto deve-se ao facto de estando o dataset equilibrado com oversampling o modelo gerado pelo classificador tem probabilidades mais próximas de identificar uma sample como negativa ou como positiva.
Sendo assim, damos preferência ao SMOTE porque é mais relevante a capacidade de identificar as minorias (positivos).

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. Por outro lado, é natural que a precision seja ligeiramente pior com SMOTE porque o modelo identifica mais vezes como positivas algumas instâncias que podem negativas enquanto que com o dataset não-equilibrado esta decisão é feita menos vezes.

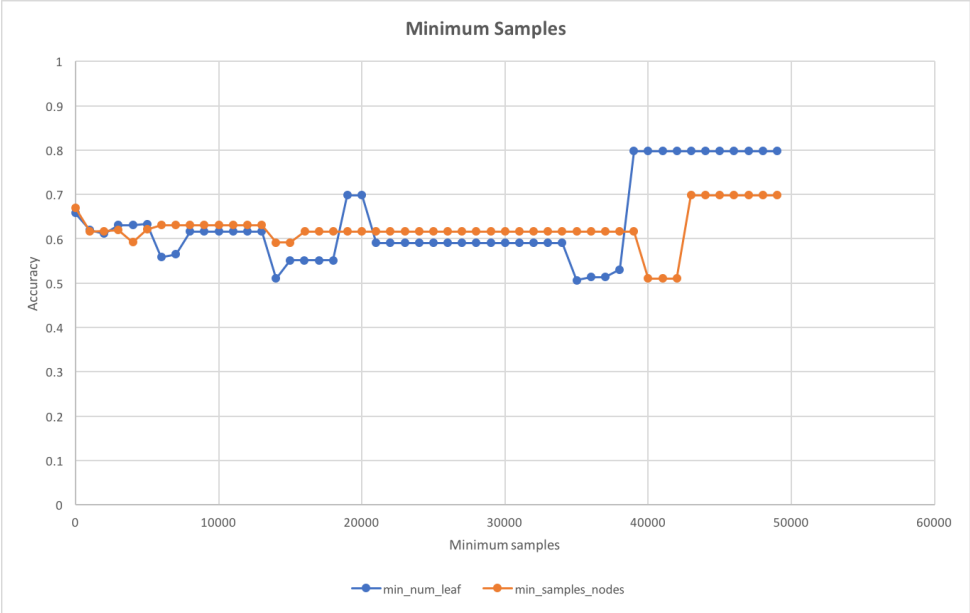


Com base nas métricas obtidas, podemos concluir que o desempenho do classificador KNN estabiliza a partir de k=21 e obtemos melhores resultados se fizermos oversampling do training set com SMOTE. A escolha de k=21 como ponto de estabilização do desempenho poderá estar relacionado com o grau de variedade média entre os atributos do dataset: por exemplo, no dataset do no-shows os atributos que têm uma amplitude maior são a idade (-1 a 110) e as datas (1 a 52) e os restantes atributos são binários.

2.3.3 Decision Trees

A análise deste dataset com Decision Trees foi similar à análise do dataset anterior, fazendo os mesmos testes, pre-pruning, variando os atributos min_samples_leaf e/ou o min_samples_split. No entanto, foi preciso balancear o dataset como foi referido em cima, e utilizámos o atributo do classificador do Python que faz o balanceamento automático do dataset com oversampling.

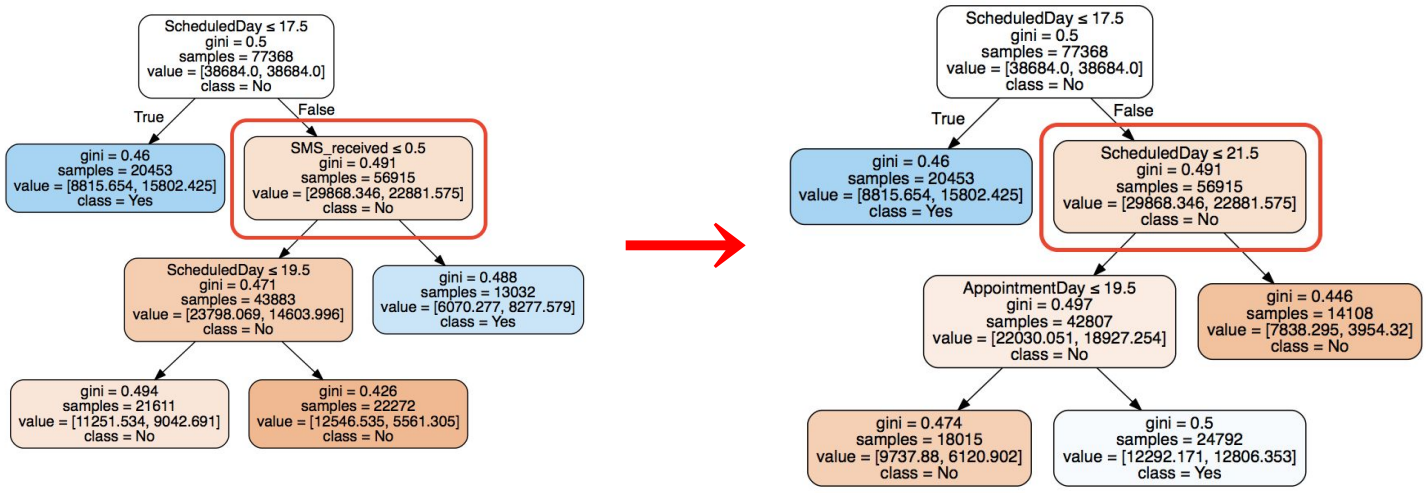
Como podemos observar, a melhor accuracy obtida com a árvore balanceada variando os dois atributos foi de 80%, mas é uma árvore de apenas um nó “No” logo os resultados são tão bons porque o dataset de teste não é balanceado (80% “No” e 20% “Yes”)
.



Neste gráfico podemos observar os acontecimentos que descrevi em cima no relatório: quando a accuracy sofre um pico de decréscimo é devido ao desaparecimento de um nível da árvore causando uma perda de informação e consequentemente o decréscimo na accuracy (exemplo nas min_num_leaf para 34000 -> 35000 amostras).

Outro acontecimento interessante é o aumento repentino da accuracy devido à mudança dos atributos a serem testados nos últimos nós da árvores: ao aumentar o número mínimo de amostras num determinado nó/folha, quando passamos o limite de um dado atributo o classificador altera o atributo a ser testado nesse nó podendo descer ou subir a accuracy.

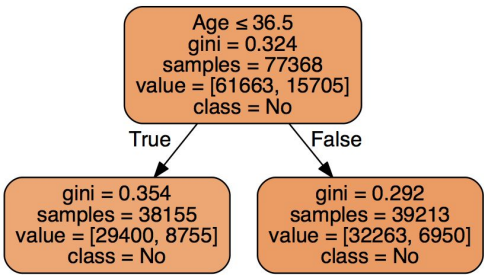
Por exemplo, nos nós com 39000- 40000 amostras mínimas, a accuracy decresce devido à mudança do atributo a ser testado (SMS_received passa para ScheduledDay).

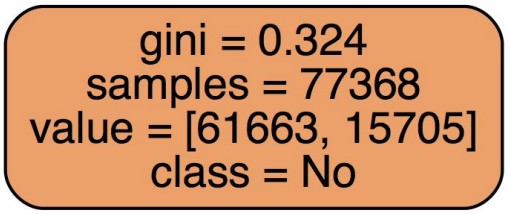


A árvore da esquerda obtém uma accuracy de aproximadamente 62% enquanto que a árvore da direita obtém quase 50%.

Facto interessante, a árvore da esquerda, que mostra ter uma accuracy de 62%, corresponde à regra que obtivemos no primeiro projecto **"ScheduledDay[18,20] => {SMS_received=0}"**.

Tal como já referi, o facto de o dataset usado para teste não ser balanceado implica que com muito pruning a certa altura obtemos árvores muito pequenas com excelentes accuracies (80 %), pois há 80% de amostras para "No" e 20% para "Yes".



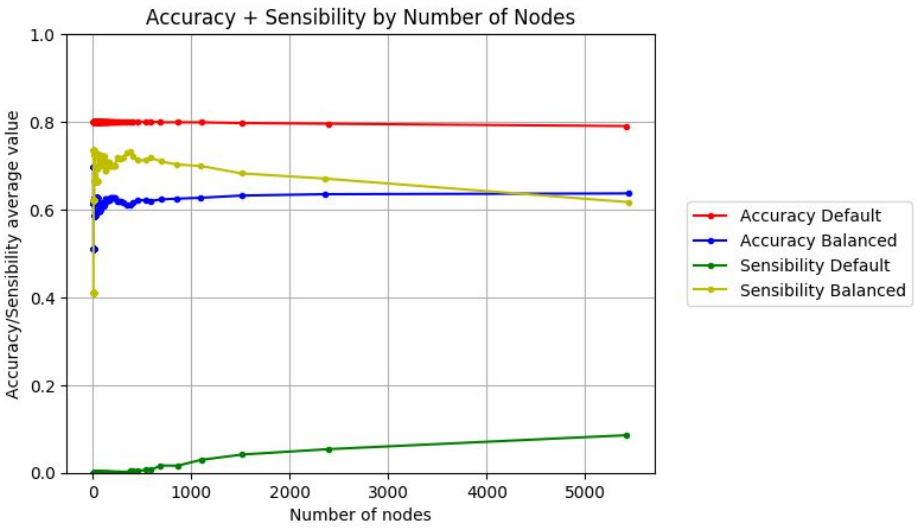


O gráfico ao lado corresponde à análise do dataset balanceado e não-balanceado, onde obtivemos os valores de accuracy e sensibility.

Podemos observar que quando não é balanceado conseguimos obter uma accuracy de quase sempre 80%. O problema é que a sensibility fica em aproximadamente 10%, porque as árvores tomam mais decisões para obter a resposta "No".

Quando balanceamos o dataset (50-50), as árvores são construídas garantindo que conseguimos obter testes que permitem chegar às duas opções "No" e "Yes" com igual probabilidade.

Por este motivo o valor de accuracy não é tão bom (60-65%), mas a sensibility aumenta bastante passando para 60-75%.



3 Critical Analysis

O algoritmo de Naive-Bayes faz a classificação de um dataset assumindo que os atributos a considerar são independentes, no entanto isto pode não ser verdade em todos os casos, pois pode haver atributos em ambos os datasets que são dependentes. Este factor leva a uma diminuição da qualidade dos resultados obtidos. Utilizámos o PCA para melhorar a distribuição da massa de probabilidades, o que conduziu a melhores resultados.

Relativamente à aplicação de KNN sobre o dataset de no-shows, para além de over-sampling também poderíamos ter usado técnicas undersampling para balancear a porção de treino. Nesse caso, os resultados e observações seriam semelhantes ao que foi obtido na classificação com Naive-Bayes: sem PCA os resultados provavelmente serão melhores do que com over-sampling e com PCA os resultados deverão ser semelhantes.

Na aplicação de over-sampling ao dataset de no-shows, escolhemos usar SMOTE por ser o método apresentado nas teóricas. No entanto, descobrimos que existe uma alternativa ao SMOTE, o ADASYN. Com ADASYN, na maioria dos casos os resultados foram semelhantes aos obtidos com SMOTE e, por isso, não achámos necessário apresentar os resultados obtidos com este método.

A nível de validação dos classificadores obtidos, também estudámos internamente a métrica de learning curve. Esta métrica permite avaliar a evolução da accuracy score com training e da accuracy com cross-validation à medida que aumentamos o número de samples disponível para training do modelo. No contexto do crabs, a learning curve para o KNN indicou que o aumento do número de samples melhora o desempenho do classificador sem PCA e piora no caso de sem PCA.

Poderíamos ainda ter feito feature selection para pré-processar os datasets, porém como já realizamos PCA em todos os casos de estudo, não sentimos necessidade de o fazer porque o PCA já faz um trabalho semelhante. Quanto aos resultados, considerámos que eliminar determinados atributos arbitrariamente, iria piorar os resultados obtidos, pois como no pré-processamento já eliminamos as colunas desprezáveis, as restantes seriam sempre necessárias para a realização da classificação.

Em relação às Decision Trees podemos verificar que em certos casos o método de pruning ajuda a melhorar os resultados, mas nem sempre.

Por exemplo, no dataset dos crabs os melhores resultados que obtivemos é quando as árvores não são pruned, visto que o dataset é pequeno com poucos atributos e está equilibrado.
No caso dos crabs verificamos que o “FL” e “CW” são sempre os primeiros atributos a serem testados, pois são os que detêm mais informação.

No caso do dataset no-shows, este tem muitos atributos e muitas amostras o que faz com que consigamos, com algum pruning, melhorar as árvores, e obter melhores resultados. Mas visto que o dataset não é balanceado, ao fazermos pruning a mais, ficamos com árvores demasiado pequenas e ao testarmos com o dataset (80-20) obtemos resultados enviesados.

Finalmente, o problema principal que encontrámos durante todo o exploração dos 2 datasets foi a falta de informação sobre a utilidade dos resultados obtidos a partir do dataset (“real life application”). Por exemplo, no caso do no-shows, era muito importante saber se é mais importante conseguir classificar sempre corretamente os positivos ou qual é a tolerância do utilizador do sistema a falsos negativos/positivos.

Outro informação que esteve em falta foi o contexto da geração do dataset e a viabilidade/confiança nos dados obtidos. Por exemplo, no dataset dos crabs, podem existir outras características que não foram registadas no dataset que permitem identificar melhor a espécie de cada crab (cor, idade, ambiente/tipo de habitat).

4 Conclusions

Com base na nossa utilização nos três métodos de classificação estudados (naive bayes, knn, decision trees) achamos que as decision trees foram a técnica mais útil porque não só era mais fácil de visualizar as decisões tomadas pelo algoritmo como também nos permitiu relacionar com os resultados obtidos na entrega anterior (regras de associação). Por outro lado embora o naive bayes seja um método simples e com pouca variação de parâmetros no caso do dataset de crabs foi um dos métodos que deu melhores resultados com utilização de PCA, variando o número de componentes.