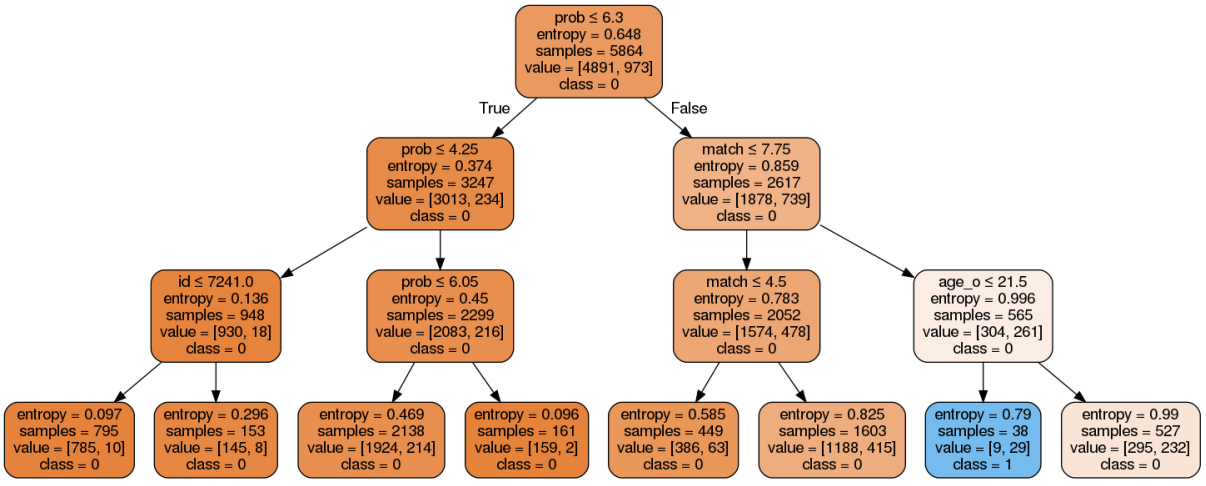
U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Criação e avaliação de modelos de classificação para um conjunto de dados recorrendo a algoritmos de Machine Learning

Árvores de Decisão (CART ou ID3) e Naive Bayes.

Unidade curricular: Inteligência Artificial CC2006



Regente da cadeira: Ana Paula Tomás

Grupo 31

Catarina Ferreira Teixeira up201805042 – **L:CC**

Patrícia Daniela Tavares Vieira up201805238 – **MI:ERSI**

JUNHO 2021

Resumo

Neste relatório avaliamos modelos de classificação de um dado conjunto de dados, que pertence a um evento experimental de speed dating da Columbia University, usando dois algoritmos de Machine Learning: Árvores de decisão (ID3) e Naive Bayes.

Para obter melhor eficiência dos algoritmos foi realizada uma análise de dados através de três estratégias, para correção de alguns elementos incorretos de acordo com os critérios pré estabelecidos no conjunto de dados fornecidos.

No sentido de procurar os melhores resultados através da comparação de valores usamos os métodos holdout e cross-validation. Através da *accuracy* e da sua matriz de confusão conseguimos obter mais informações que foram fundamentais para melhor avaliação dos métodos.

Introdução ao problema

Dado um evento experimental de speed dating, entre 2002 e 2004, pela Columbia University, vamos criar e avaliar modelos de classificação destes dados adquiridos recorrendo a algoritmos de Machine Learning.

Neste evento os participantes foram recrutados entre os estudantes da universidade, onde teriam um "primeiro encontro" de quatro minutos com todos os outros participantes, ao longo e no final do encontro eram questionados. Neste trabalho foi selecionado um subconjunto de atributos recolhidos ao longo da experiência, precisamente 8378 registos.

Começamos por dividir o trabalho em duas etapas, o tratamento de dados e o desenvolvimento de algoritmos, para melhor prestabilidade do projeto.

Na primeira etapa principiamos por ler os dados e analisá-los, de seguida prosseguimos para o pré-processamento dos mesmos.

Por último, para cada algoritmo que iremos utilizar, vamos estudar os parâmetros a usar e avaliar os algoritmos escolhidos e no final apresentar os resultados. Caso os resultados não sejam os pretendidos, voltamos ao passo inicial da segunda etapa, ao estudo dos parâmetros.

Para a implementação dos algoritmos optamos por usar a linguagem Python, dada a grande quantidade de documentação online e bibliotecas de Machine Learning disponíveis para esta linguagem. As bibliotecas que usamos foram pandas, scikit-learn, numpy, six, ipython e pydotplus.

Algoritmos

Para criar e avaliar modelos de classificação para o conjunto de dados obtidos, usamos dois algoritmos de Machine Learning:

- Árvores de decisão - **ID3**;
- **Naive Bayes**.

No sentido de procurar a melhor eficiência de cada algoritmo, utilizamos três estratégias distintas. Ao encontrar campos com valores NaN (valores nulos), temos a opção de eliminar toda a linha referente a esse valor, substituir esse valor por zero ou alterar esse valor para a média ou moda (dependendo dos casos) da coluna onde foi encontrado. Estas estratégias vão ser usadas nos algoritmos seguintes e serão melhor explicadas no tópico seguinte.

ID3

A árvore de decisão é representada por uma função que recebe como input um conjunto de dados de acordo com a estratégia que definimos. Esta vai retornar uma decisão como output que verifica se foram previstos como match (1) ou como não match (0).

Para conseguir uma avaliação correta, este tem que ser executado, consecutivamente.

Cada nó da árvore vai ser representado como um atributo dado, e cada ramo dela vai representar um valor possível para esse atributo.

O algoritmo ID3 irá ser representado da seguinte forma:

function DECISION-TREE-LEARNING($examples$, $attributes$, $parent\_examples$) **returns** a tree

if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$)
else if all $examples$ have the same classification **then return** the classification
else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, $examples$)
  $tree \leftarrow$ a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples \text{ and } e.A = v_k\}$
    $subtree \leftarrow$ DECISION-TREE-LEARNING(exs, $attributes - A$, $examples$)
    add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
  return $tree$

PLURALITY-VALUE - é o valor de output mais comum nos exemplos dados, e em caso de empate, vai selecionar aleatoriamente o resultado;
IMPORTANCE - este vai selecionar o atributo, que vai ter mais exemplos, ajuda a classificar com o objetivo de adquirir a menor altura da árvore possível.

Esta função é implementada através da **noção de ganho de informação,** que é baseado no conceito da entropia. A entropia de uma variável aleatória V, com valores v_k, cada um com probabilidade $P(v_k)$, é dado por:

$$H(V) = -\sum_k P(v_k) log_2 P(v_k)$$

A partir da entropia, queremos saber qual será o ganho de informação que vamos ter ao escolhermos um atributo A, como o próximo atributo a ser testado para classificar um exemplo.

Primeiro, precisamos de saber a entropia no conjunto de dados gerado pela divisão imposta pela escolha do atributo A.

Dado um conjunto E de exemplos e um atributo A dado, necesitamos de calcular $H(E|A)$. Subtraímos os dois valores de entropia (antes e depois da escolha de A), obtemos qual é a redução do valor de entropia a partir de A - vamos obter o valor do ganho de informação.

$$H(E|A) = \sum_{s \in S} p(s) H(s)$$

S é o conjunto de subconjuntos dos exemplos de E, agrupados de acordo com o seu valor para o valor A.

A partir disso, vamos definir o ganho de informação, que é definido por:

$$IG(E, A) = H(E) - H(E|A)$$

E é o conjunto de dados inicial e E_i o subconjunto de E cujo exemplos têm o valor i para o atributo A.

Naive Bayes

O algoritmo Naive Bayes incorpora o teorema de Bayes, que prever resultados a partir de modelos probabilísticos. O Teorema de Bayes é baseado no modelo em que dado dois eventos X e Y, temos que:

$$P(Y|X) = \frac{P(X|T)P(Y)}{P(X)}$$

em que:
-$P(X)$ e $P(Y)$ são as probabilidades a priori de X e Y;
-$P(X|Y)$é a probabilidade a posteriori ou condicional de X dado Y;
-$P(Y|X)$é a probabilidade a posteriori ou condicional de Y dado X.

O teorema mostra como alterar as probabilidades a priori, $P(X)$e $P(Y)$, tendo em conta novas evidências $P(X|Y)$de forma a obter probabilidades a posteriori $P(Y|X)$.

Embora esta regra seja ótima, a sua aplicabilidade é reduzida devido ao grande número de exemplos necessários para calcular, de forma fiável, $P(X|Y)$.

Assumindo que existe uma função objetivo $f: X \rightarrow Y$onde cada instância x é descrita por p atributos $< x_1, x_2,..., x_p >$

O valor mais provável de $f(x)$é a classe $y_j \in Y$que maximiza $P(y_j|x_1, x_2,..., x_p)$

$$\begin{aligned}
\hat{y} &= \underset{y_j \in Y}{\operatorname{argmax}} P(y_j|x_1, x_2 \dots x_p) \\
\hat{y} &= \underset{y_j \in Y}{\operatorname{argmax}} \frac{P(x_1, x_2 \dots x_p|y_j)P(y_j)}{P(x_1, x_2 \dots x_p)} \\
&\propto \underset{y_j \in Y}{\operatorname{argmax}} P(x_1, x_2 \dots x_p|y_j)P(y_j)
\end{aligned}$$

Assunção de independência

Uma das hipóteses para a assunção de independência é assumir que cada atributo é condicionalmente independente dos outros, dada a classe.

$$P(x_1, x_2 \dots x_n|y_j) = \prod_i P(x_i|y_j)$$

A previsão é obtida da forma:

$$\hat{y} = \underset{y_j \in Y}{\operatorname{argmax}} P(y_j) \prod_i P(x_i|y_j)$$

A classe para a qual for obtido um valor mais alto de probabilidade, é a classe atribuída ao exemplo → o Naive Bayes é um classificador probabilístico.

O modelo que usamos para estimar as probabilidades é o Gaussian Naive Bayes. Ele é definido da seguinte maneira:

- Para um dado atributo numérico X_k, a probabilidade de observar um valor $x_k$dada uma classe y_j, assumindo uma distribuição Normal é dada por

$$P(x_k|y_j) = \frac{1}{\sigma_{kj}\sqrt{2\pi}} e^{-\frac{(x_k - \mu_{kj})^2}{2\sigma_{kj}^2}}$$

onde μ_{kl} e σ_{kj}são, respetivamente, a média e o desvio padrão dos valores do atributo $X_k$nos exemplos cuja classe é y_j.

Análise exploratória dos dados

No subconjunto que nos foi atribuído para a elaboração deste trabalho, os 8378 registos encontravam-se divididos por 13 campos/atributos. Ao ler e analisar estes valores podemos retirar algumas informações fundamentais para o *data cleaning*.

Uma das primeiras informações que tiramos foi que 12 desses campos correspondem a perguntas/dados que estavam contidas nos questionários e em cada uma desses podemos expor que:

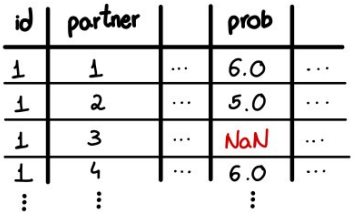
- **id** - número de identificação do participante;
- **partner** - número de identificação do par;
- **age** - idade do participante:
  - os valores situam-se num intervalo de 18 a 55;
  - a média das idades é 26 anos;
- **age_o** - idade do par:
  - os valores situam-se num intervalo de 18 a 55;
  - a média das idades é 26 anos;
- **goal** - "Qual é o seu objetivo principal ao participar neste evento?":
  - NaN (0.9%);
  - 1- Passar uma noite divertida (40.9%);
  - 2- Conhecer novas pessoas (36%);
  - 3- Conseguir um encontro (7.5%);
  - 4- Procurar um relacionamento sério (3.6%);
  - 5- Dizer que consegui (6.1%);
  - 6- Outro (5%) ;
- **date** - "Em geral, quão frequentemente sai para encontros?":
  - NaN (1.2%);
  - 1- Várias vezes por semana (1.1%);
  - 2- Duas vezes por semana (3.7%);
  - 3- Uma vez por semana (9.3%);
  - 4-Duas vezes por mês (24.3%);
  - 5- Uma vez por mês (18.2%);
  - 6- Várias vezes por ano (25%);
  - 7- Quase nunca (17.2%);
- **go_out** - "Com que frequência sai (não necessariamente para encontros)?":
  - NaN (1.0%);
  - 1- Várias vezes por semana (31.1%);
  - 2- Duas vezes por semana (35.7%);
  - 3- Uma vez por semana (23.3%);
  - 4- Duas vezes por mês (5.4%);
  - 5- Uma vez por mês (2%);
  - 6- Várias vezes por ano (1.1%);
  - 7- Quase nunca (0.4%);
- **int_corr** - "Correlação entre os ratings de interesses (desporto, museus, caminhadas, música, filmes, livros, etc.) do participante e do seu par ([-1,1])";
- **length** - "A duração de 4 minutos para o encontro é:":
  - NaN (11%);
  - 1- Demasiado curta (50.4%);
  - 2- Demasiado longa (2.1%);
  - 3- Adequada (36.5%);
- **met** - "Já conhecia o seu par anteriormente? (0/1)":
  - NaN (4.5%), Não (48.3%) e Sim (47.2%);
- **like** - "Quão gostou do seu par? (escala 1-10; nada = 1; muito =10)":
  - NaN, (2.9%),0-5 (32.2%) e 5.5-10 (64.9%);
- **prob** - "Qual é a probabilidade do seu par ter gostado de si? (escala 1-10; pouco provável = 1; muito provável =10)":
  - NaN (3.7%), 0-5 (53.1%), 5.5-10 (43.2%).

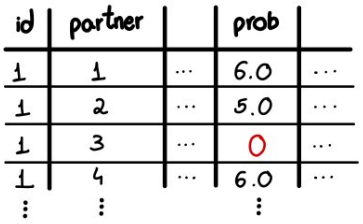
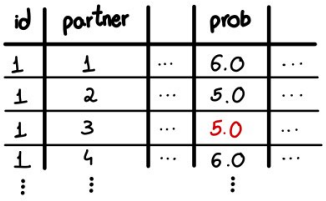
O último campo não se encontra nos questionários, mas sim foi inserido pelos desenvolvedores do estudo de acordo com os seus modelos de classificação dos dados, campo esse no nosso estudo vai servir para variável objetivo.

- **match** - "Há match? (0/1)?":
  - Não (16%) e Sim (84%)

Depois de ler os dados e analisá-los, passamos para o pré-processamento dos registos, nesta fase de *data cleaning* tentamos identificar se existiam valores incorretos ou NaN entre os registos.

Deparamos-nos com mais campos com valores **NaN** e neste caso usamos 3 abordagens/estratégias diferentes para futuramente realizar comparações nos algoritmos:

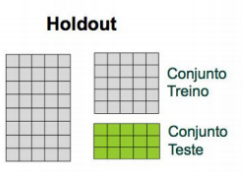


eliminar toda a linha	substituir por zero	substituir para média, moda ou mediana
Se na linha n tiver algum campo **NaN**, essa linha é eliminada dos registos	Todos os campos com valor **NaN** são substituídos por 0	Dependendo dos atributos substituímos os valores pela média, moda ou mediana com base nos restantes valores da coluna.
		

Estas abordagens vão servir para fazer comparações entre algoritmos, para verificar em qual dos casos vai ser mais eficiente.

Por cada estratégia foram usados dois métodos:

- método **holdout**
- método **Cross-Validation**



Método holdout

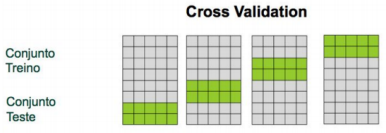
O método **holdout** vai dividir aleatoriamente o nosso conjunto de dados em treino/teste (tipicamente, **70%/30%**);

Método Cross-Validation

Dividimos o conjunto de dados em **k partições/folds** (neste caso dividimos em 10).

Usamos alternadamente, cada partição/fold como conjunto de teste e as restantes **k-1 partições/folds** como conjunto de treino.

Todos os exemplos realizados vão ser usados para treino e para o teste. No final fazemos a média da avaliação em cada uma das **k iterações**.



Para descobrir que casos foram previstos corretamente e os que não foram previstos corretamente, utilizamos o que é chamado de matriz de confusão.

E com isso podemos obter várias métricas:

- **precision** = TP / (TP+FP)
- **recall** = TP / (TP+FN)
- **....**

		Predicted Class	
		P	*N*
True Class	*P*	*TP* True Positive	*FN* False Negative
	N	*FP* False Positive	*TN* True Negative

Experiência e resultados

Para cada estratégia resolvemos pelo método holdout e o método cross-validation. Descobrimos também o *accuracy*, a sua matriz de confusão e o seu classification report, que nos dá informação sobre o precision, recall, etc…

Para os resultados apresentamos um resultado que vai refletir aproximadamente ao output de vários testes.

Eliminar toda a linha

Sucintamente, esta estratégia vai eliminar a linha se houver a ocorrência de um **NaN**, o que vai reduzir drasticamente o número de linhas da base de dados (passa de 8378 para 6878). Depois fizemos a transformação para valores inteiros de todas as colunas exceto o **prob**, o **like**, o **int_corr** e o **match**, e de seguida resolvemos consoante o método.

O ficheiro que se encontra a resolução deste ficheiro é o **drop.ipynb** ou então se quiser executar individualmente em python têm estes ficheiros: **dropID3holdout.py, dropID3cross.py, dropNBholdout.py, dropNBcross.py.**

ID3

Método holdout (resultado):

Accuracy: 0.7650193798449613

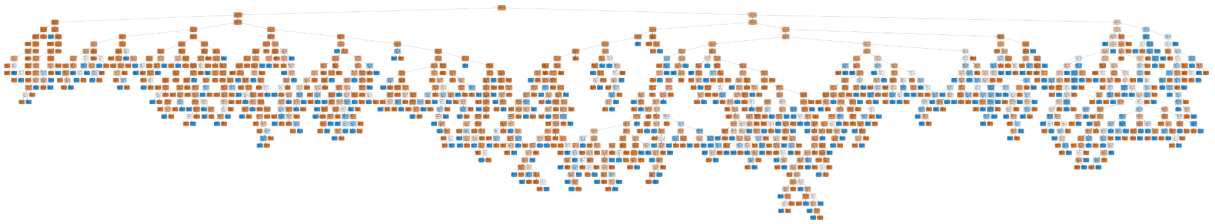
Matriz de confusão:

Real \ Previsto	0	1
0	1450	245
1	240	129

Classification report:

	precision	recall	f1-score	support
0	0.86	0.86	0.86	1695
1	0.34	0.35	0.35	369
accuracy			0.77	2064
macro avg	0.60	0.60	0.60	2064
weighted avg	0.77	0.77	0.77	2064

Árvore de decisão



Método Cross-Validation (resultado):

Array de accuracies: [0.74854651, 0.68023256, 0.71947674, 0.67151163, 0.74563953, 0.71511628, 0.68168605 0.68459302, 0.69286754, 0.72634643]

Média de accuracy: 0.7066016299380522

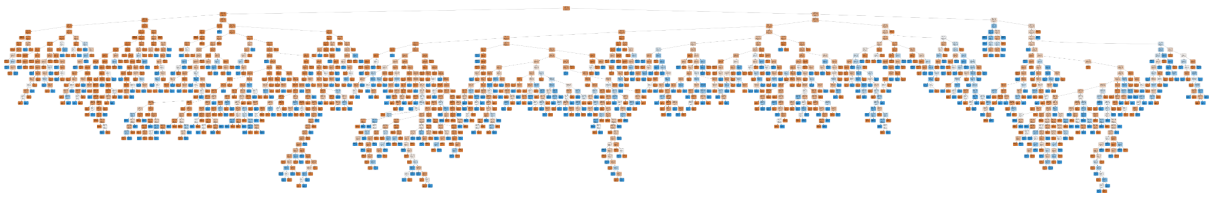
Matriz de confusão para cada partição/fold:

```
[[407 179] [[448  94] [[500  76] [[445 110] [[468 102]
 [ 54  48]] [ 91  55]] [ 86  26]] [ 88  45]] [ 85  32]]
[[493  62] [[479  97] [[489 115] [[455 104] [[417 161]
 [100  33]] [ 83  29]] [ 56  28]] [ 94  35]] [ 71  38]]
```

Soma da matriz de confusão com todos as partições:

Real \ Previsto	0	1
0	4601	1100
1	808	369

Árvore de decisão



Naive Bayes

Método holdout (resultado):

Accuracy: 0.8471615720524017

Matriz de confusão:

Real \ Previsto	0	1
0	567	11
1	94	15

Classification report:

	precision	recall	f1-score	support
0	0.86	0.98	0.92	578
1	0.58	0.14	0.22	109
accuracy			0.85	687
macro avg	0.72	0.56	0.57	687
weighted avg	0.81	0.85	0.81	687

Método Cross-Validation (resultado):

Array de accuracies: [0.81540698, 0.81395349, 0.83430233, 0.74563953, 0.84156977, 0.7994186, 0.8372093, 0.80232558, 0.81222707, 0.83842795]

Média de accuracy:0.814048060322941

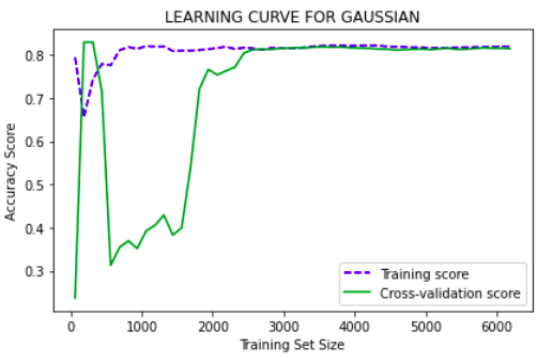
Matriz de confusão para cada partição/fold:

```
[[517  69]  [[511  31] [[563  13] [[542  13]  [[526  44]
 [ 60  42]]  [102  44]] [ 94  18]] [113  20]]  [ 87  30]]
[[524  31]  [[482  94] [[554  50] [[506  53]  [[567  11]
 [101  32]]  [ 74  38]] [ 65  19]] [ 82  47]]  [ 94  15]]
```

Soma da matriz de confusão com todos as partições:

Real \ Previsto	0	1
0	5292	409
1	872	305

Curva de aprendizagem:



Substituir por zero

Esta estratégia, de maneira sucinta, vai substituir todos os valores NaN por zero. Depois de fazer a transformação para inteiro de todas as colunas exceto o **prob**, o **like**, o **int_corr** e o **match**, prosseguimos para a resolução do método.

O ficheiro que se encontra a resolução deste ficheiro é o **fill0.ipynb** ou então se quiser executar individualmente em python têm estes ficheiros: **fill0ID3holdout.py, fill0ID3cross.py, fill0NBholdout.py, fill0NBcross.py.**

ID3

Método holdout (resultado):

Accuracy: 0.7836117740652346

Matriz de confusão:

Real \ Previsto	0	1
0	1831	276
1	268	139

Classification report:

	precision	recall	f1-score	support
0	0.87	0.87	0.87	2107
1	0.33	0.34	0.34	407
accuracy			0.78	2514
macro avg	0.60	0.61	0.60	2514
weighted avg	0.79	0.78	0.78	2514

Árvore de decisão

Método Cross-Validation (resultado):

Array de accuracies: [0.73627685, 0.69570406, 0.71599045, 0.68973747, 0.74940334, 0.74224344, 0.76968974, 0.63484487, 0.7311828, 0.77538829]

Média de accuracy: 0.7240461302013383

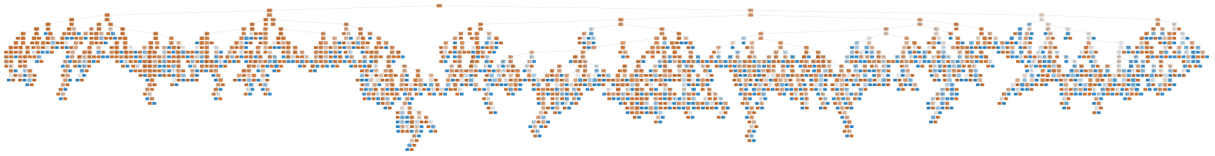
Matriz de confusão para cada partição/fold:

```
[[554 155] [[560 117] [[599 109] [[593  96] [[618  87]
 [ 79  50]] [106  55]] [ 90  40]] [107  42]] [108  24]]
[[585  93] [[544 162] [[580 148] [[550 131] [[585 132]
 [109  51]] [ 84  48]] [ 72  38]] [110  47]] [ 89  31]]
```

Soma da matriz de confusão com todos as partições:

Real \ Previsto	0	1
0	5768	1230
1	954	426

Árvore de decisão



Naive Bayes

Método holdout (resultado):

Accuracy:0.8649940262843488

Matriz de confusão:

Real \ Previsto	0	1
0	710	7
1	106	14

Classification report:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.99      0.93       717
           1       0.67      0.12      0.20       120

    accuracy                           0.86       837
   macro avg       0.77      0.55      0.56       837
weighted avg       0.84      0.86      0.82       837
```

Método Cross-Validation (resultado):

Array de accuracies: [0.79713604, 0.81264916, 0.83770883, 0.78520286, 0.8353222, 0.81264916, 0.84844869, 0.79116945, 0.81242533, 0.84468339]

Média de accuracy: 0.8177395117806234

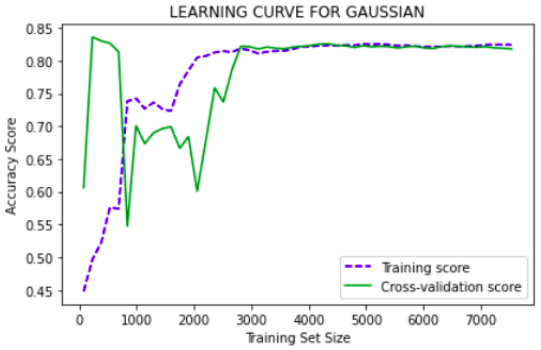
Matriz de confusão para cada partição/fold:

```
[[604 105] [[638  39] [[691  17] [[677  12] [[660  45]
 [ 70  59]] [113  48]] [112  18]] [124  25]] [110  22]]
[[641  37] [[606 100] [[686  42] [[602  79] [[710   7]
 [121  39]] [ 86  46]] [ 96  14]] [104  53]] [106  14]]
```

Soma da matriz de confusão com todos as partições:

Real \ Previsto	0	1
0	6515	483
1	1042	338

Curva de aprendizagem:



Substituir para média, moda ou mediana

Esta vai ser a estratégia principal, onde vamos preencher os dados que estão com **NaN** e fazer algumas alterações ao conjunto de dados.

Primeiro, verificamos que existe um **id** a **NaN** na última posição do conjunto de dados, verificamos também que está ordenado de forma crescente de id/partner, então assumimos logo que podemos substituir esse **id** com o valor **22**.

Também verificamos, que na coluna **prob** e **like** existem valores com **0.0** quando o seu range é de **1-10**. Assumimos então que se deu **0** foi porque não gostou nada, para ir de encontro com o nosso range, substituímos todos os nossos **0.0** por **1.0**.

Para tratar do **NaN** em ambas as colunas, pois ambos os atributos são **categóricos ordinais**, normalmente trata-se este atributo como **numérico** e o processo mais correto para tratar destes valores é com a **média**, então vamos preencher todos os **NaN** com a **média** do **prob** e com a **média** de **like** respetivamente.

Para preencher a idade (**age** e **age_o**) como os atributos são **numéricos discretos,** normalmente trata-se destes valores com a **média** ou com a **mediana**. Neste caso, decidimos usar a **mediana** para preencher o **NaN** em ambas as colunas.

Nas restantes colunas, utilizamos a moda para preencher o **NaN**, porque os atributos são **categóricos,** e para atributos **categóricos** utiliza-se a **moda** para o preenchimento.

Depois de todas estas modificações e de interpretar o conjunto de dados, verificamos que a coluna **int_corr** não seria relevante para avaliação dos modelos, porque não iria influenciar em nada no processo de matching.

De seguida, fizemos a transformação para valores inteiros de todas as colunas exceto o **prob**, o **like**, o **int_corr** e o **match** e resolvemos o método.

O ficheiro que se encontra a resolução deste ficheiro é o **principal.ipynb** ou então se quiser executar individualmente em python têm estes ficheiros: **principalID3holdout.py, principalID3cross.py, principalNBholdout.py, principalNBcross.py.**

ID3

Método holdout (resultado):

Accuracy: 0.7939538583929993

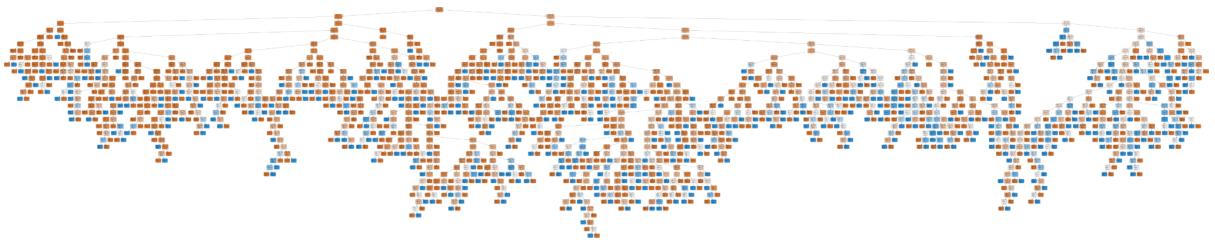
Matriz de confusão:

Real \ Previsto	0	1
0	1837	270
1	248	159

Classification report:

	precision	recall	f1-score	support
0	0.88	0.87	0.88	2107
1	0.37	0.39	0.38	407
accuracy			0.79	2514
macro avg	0.63	0.63	0.63	2514
weighted avg	0.80	0.79	0.80	2514

Árvore de decisão



Método Cross-Validation (resultado):

Array de accuracies: [0.70167064, 0.72911695, 0.6849642, 0.59665871, 0.74940334, 0.72673031, 0.78162291, 0.58353222, 0.72998805, 0.7562724]

Média de accuracy: 0.7039959738011936

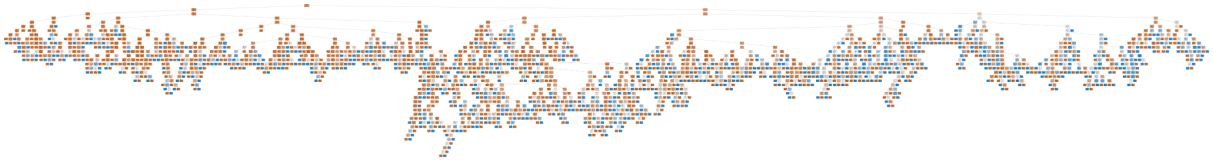
Matriz de confusão para cada partição/fold:

```
[[555 154]  [[507 170]  [[616  92]  [[586 103]  [[586 119]
 [ 80  49]]  [ 97  64]]  [101  29]]  [100  49]]  [ 95  37]]
[[596  82]  [[575 131]  [[555 173]  [[520 161]  [[633  84]
 [115  45]]  [ 92  40]]  [ 69  41]]  [ 97  60]]  [ 93  27]]
```

Soma da matriz de confusão com todos as partições:

Real \ Previsto	0	1
0	5729	1269
1	939	441

Árvore de decisão



Naive Bayes

Método holdout (resultado):

Accuracy: 0.8661887694145759

Matriz de confusão:

Real \ Previsto	0	1
0	709	8
1	104	16

Classification report:

	precision	recall	f1-score	support
0	0.87	0.99	0.93	717
1	0.67	0.13	0.22	120
accuracy			0.87	837
macro avg	0.77	0.56	0.57	837
weighted avg	0.84	0.87	0.83	837

Método Cross-Validation (resultado):

Array de accuracies: [0.79713604, 0.81980907, 0.849642, 0.78639618, 0.83651551, 0.81742243, 0.85441527, 0.79713604, 0.8255675, 0.84348865]

Média de accuracy: 0.8227528706626405

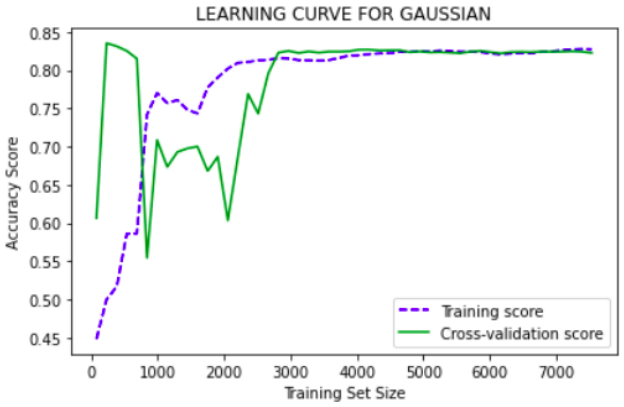
Matriz de confusão para cada partição/fold:

```
[[614  95] [[641  36] [[689  19] [[676  13] [[666  39]
 [ 69  60]] [104  57]] [112  18]] [119  30]] [109  23]]
[[644  34] [[610  96] [[692  36] [[610  71] [[709   8]
 [123  37]] [ 86  46]] [ 97  13]] [104  53]] [104  16]]
```

Soma da matriz de confusão com todos as partições:

Real \ Previsto	0	1
0	6551	447
1	1027	353

Curva de aprendizagem:



Discussão dos resultados

Depois de executarmos, verificamos que a terceira estratégia é mais eficiente e correta em relação às outras duas (**eliminar** ou **preencher por 0**) por estas razões:

- Iremos ter mais dados para testar no programa que elimina as linhas com os **NaN**, sendo que o programa que tem menos dados a fazer o **holdout** e o **cross-validation** o conjunto teste pode ser muito pequeno para fazer uma estimativa.
- Na estratégia que preenche todos os **NaN** com **0** é muito incorreta porque há colunas em que não pode ter valores **0**, como por exemplo o **id**, **age**, **age_o**, **prob**, **like**. Isso irá afetar a eficiência dos modelos.

A executar todos os programas podemos concluir que, o método **holdout** é o que irá dar a melhor eficiência em ambos os modelos (**ID3** e **Naive Bayes**) devido ao facto que está a testar um conjunto único de teste e treino em vez de testar todas as partições, o que poderá haver partições muito piores que outras e a fazer a média final pode prejudicar em muito a sua eficiência.

Também reparamos que a curva de aprendizagem irá ser melhor na nossa estratégia sendo que inicialmente ele está longe de estar perto da curva de treino, mas depois no final ele já se aproxima muito dele, concluindo que ele foi bem treinado para prever o resultado.

Verificando a **matriz de confusão** de todas as execuções podemos concluir que o nosso conjunto de dados não é estável no **match**, porque vamos ter muito mais **não match (0)** que **match (1)**, logo o algoritmo vai ser melhor para prever qual não deu match do que deu match. Tal podemos verificar fazendo a **média** para descobrir a percentagem de **match (1)** e **1 menos a média** para descobrir a percentagem de no **match (0)**, sendo que cerca de **16,5%** de **match** e **83,5%** de **não match (0)**.

Limitações do trabalho e dificuldades

Neste trabalho as limitações que mais tivemos foi primeiramente saber trabalhar muito bem com o python e as suas bibliotecas para realizar este trabalho, porém com algum trabalho e pesquisa conseguimos nos adaptar.

Demoramos também algum tempo para interpretar o conjunto de dados e perceber os algoritmos de machine learning e como é que conseguiríamos chegar a um resultado, mas depois de alguma pesquisa sobre as bibliotecas e como executar os algoritmos lá conseguimos chegar a uma boa estratégia que gerou uma boa eficiência.

Conclusão

Ao desenvolver este projeto assimilamos conhecimentos relacionados com algoritmos de Machine Learning, mais especificamente ID3 e Naive Bayes. Através da análise destes algoritmos e estratégias diferentes analisamos que com o mesmo conjunto de dados conseguimos adquirir valores/resultados bastantes diferentes.

Apesar de todas as limitações que fomos encontrando ao longo do trabalho, conseguimos saber interpretar, analisar e limitar todos os parâmetros de forma a atingir os melhores valores para os algoritmos usados.

Uma das conclusões que retiramos da análise de dados é que a melhor estratégia, neste projeto, para a correção de erros é substituir os valores NaN pela média, moda ou mediana de acordo com a coluna/campo e ao encontrar valores que não se encontram no intervalo, alterá-los de acordo com os casos.

Somos também capazes de afirmar que o método holdout é o que dará o melhor resultado em ambos os modelos utilizados.

De acordo com todos os resultados que nos foram devolvidos dos algoritmos e das suas variantes, podemos concluir que o melhor resultado que nos foi dado, foi o algoritmos Naive Bayes com o método holdout, com accuracy de 0.866.

Referências bibliográficas

Slides da cadeira;
https://machinelearningmastery.com/k-fold-cross-validation/
https://pychill.info/algorithms/build-a-decision-tree-using-id3-algorithm-with-python/
https://www.geeksforgeeks.org/using-learning-curves-ml/
https://www.dataquest.io/blog/learning-curves-machine-learning/
https://in.springboard.com/blog/naive-bayes-classification/
https://towardsdatascience.com/decision-tree-from-scratch-in-python-46e99dfea775
https://www.datacamp.com/community/tutorials/decision-tree-classification-python
https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn
https://medium.com/@lope.ai/decision-trees-from-scratch-using-id3-python-coding-it-up-6b79e3458de4
https://stackabuse.com/decision-trees-in-python-with-scikit-learn
https://www.python-course.eu/Decision_Trees.php