

FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO

# INTELIGÊNCIA ARTIFICIAL

## MACHINE LEARNING

Grupo 47

Alunos:

Brenda Karoline Cruz Nogueira (201701045)-PL5

Igor Gabriel Soares Mélo (201900603)-PL4

# RESUMO

O relatório descreve a análise dos dados e resultados baseados em estudo realizado pela Columbia University que tinha como objetivo o uso de Speed Dating para marcação de encontros. Foram desenvolvidos e avaliados modelos de classificação para análise desses dados e, dessa forma, descobrir as tendências de comportamento que levam ao *match* e, conseqüentemente, ao não *match*.

Para isso, foi necessário o uso de dois algoritmos, a Árvore de Decisão (CART) e o *Naive Bayes*. A linguagem *Python* foi a escolhida para o desenvolvimento do projeto em conjunto com a ferramenta *Colaboratory*, do Google, para compilar e testar o código de forma que ambos os integrantes tivessem acesso simultâneo ao desenvolvimento do trabalho. A linguagem R foi utilizada para auxiliar na análise dos dados.

Os resultados obtidos revelaram que tópicos como a idade do par e o motivo de usar o Speed Dating não costumam influenciar no *match*. Já tópicos como *Prob*, *like* e *date* da entrevista, tiveram certa influência no possível *match*. Dessa forma, a pesquisa mostrou que características muito específicas não costumam aumentar o índice de matches tendo em vista a distribuição quase normal da maioria dos tópicos. Por fim, apesar dos modelos em questão de acurácia não serem muito eficazes, comparados a um *dummy model*, ainda foi possível obter resultados interessantes o que permitiu realizar outros tipos de análise como a precisão e o recall, além de analisar o F1-Score, dessa forma concluiu-se que dentre os modelos analisados o Naive Bayes se mostrou mais eficaz nesses parâmetros.

# INTRODUÇÃO

Com base nos dados de um estudo realizado pela Columbia University sobre o uso de *Speed Dating* para marcação de encontros, foi desenvolvido esse trabalho com o objetivo de, com o auxílio de algoritmos de análise de dados, entender e conseqüentemente tentar prever o comportamento dos participantes. Para isso, o trabalho foi baseado na linguagem Python que conta as bibliotecas Pandas (para análise de dados) e a biblioteca *Scikit-learn* que conta com algoritmos de aprendizagem supervisionada, e foi utilizada a linguagem R para auxiliar na análise dos dados. Foi feito o uso de aprendizagem supervisionada e baseada em modelos como *Batch Learning*, além do uso da Árvore de Decisão (CART) e o *Naive Bayes*. O uso desses métodos e algoritmos permitiu chegar a resultados e conclusões interessantes quanto ao comportamento das pessoas envolvidas na pesquisa.

# ALGORITMOS

Com o objetivo de avaliar modelos de classificação para o conjunto de dados apresentados, recorreremos a dois algoritmos: CART e *Naive Bayes*.

O CART é um modelo baseado em procura em uma árvore de decisão, no qual é realizado testes lógicos nos nós internos e os nós folha possuem a classe a que determinado caminho percorrido pela árvore corresponde, no caso de ser uma árvore de classificação, como é o nosso caso.

Os nós internos são escolhidos de forma a melhor divide o conjunto de dados, no caso do CART o critério usado em uma árvore de classificação é a **impureza de Gini**. Logo, é escolhido o nó que maximiza a redução da impureza de Gini, ou seja, que maximiza s:

$$s = \text{Gini}(t) - \text{Gini}(s, t)$$

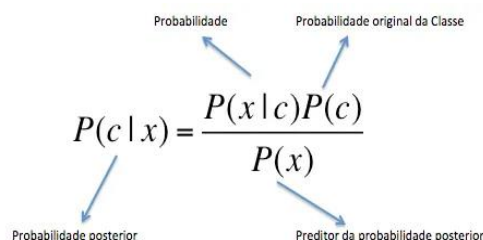
$$\text{No qual } t \text{ é um certo nó e } \text{Gini}(s, t) = \frac{n_l}{n_t} \times \text{Gini}(t_l) + \frac{n_r}{n_t} \times \text{Gini}(t_r) \text{ e } \text{Gini}(t) = 1 - \sum_{i=1}^c p_i^2.$$

Onde  $p_i$  representa a frequência dos resultados pertencerem a classe  $i$  sobre o número total resultados, em um conjunto de dados no nó  $t$ , e  $n_l$  é o número de dados que estão no filho esquerdo,  $n_r$  o número de dados que estão no filho direito e  $n_t$  o número total de dados no nó pai. Logo, devemos comparar a impureza de Gini antes e depois da expansão de um nó.

Para dividir os dados, se o atributo for numérico ele começa por ordenar os dados e realiza testes para todos os pontos intermédios. O teste que obtiver melhor desempenho será o que será usado na árvore para a variável. Já em atributos categóricos avalia todos os subconjuntos possíveis do conjunto de valor e o que melhor dividir os dados será usado para a variável.

Já o algoritmo *Naive Bayes* é um classificador probabilístico baseado no “Teorema de Bayes”, tem como qualidades a sua simplicidade e a rapidez, o que resulta em um bom desempenho em aplicações de *Machine Learning*, além de precisar de uma pequena quantidade de dados de teste para concluir classificações com boa precisão.

A sua principal característica se dá pelo fato dele desconsiderar completamente a correlação entre as variáveis, assim, o tratamento das mesmas se dá de forma independente. Teorema de Bayes fornece uma forma de calcular a probabilidade posterior  $P(C | X)$  a partir de  $P(C)$ ,  $P(x)$  e  $P(X | c)$ .


$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

Onde  $P(c | x)$  é a probabilidade posterior da classe ( $c$ , alvo) dada preditor ( $x$ , atributos),  $P(c)$  é a probabilidade original da classe,  $P(x | c)$  é a probabilidade que representa a probabilidade de preditor dada a classe e  $P(x)$  é a probabilidade original do preditor. Para atributos categóricos, a probabilidade é estimada a partir de tabelas de frequência, já para atributos numéricos podemos assumir uma distribuição normal (gaussiana), de Bernoulli, Multinomial, etc.

# ANÁLISE EXPLORATÓRIA DOS DADOS

Iniciou-se utilizando o *Python* para a importação dos dados e pré-processamento dos mesmos. Primeiramente, foi retirada a primeira coluna indefinida do conjunto de dados, e depois foi requisitada uma descrição dos dados. A tabela a seguir mostra a primeira linha.

	id	partner	age	age_o	goal	date	go_out	int_corr	length	met	like	prob	match
count	8377.00 0000	8378.00 0000	8283.00 0000	8274.00 0000	8299.00 0000	8281.00 0000	8299.00 0000	8220.00 0000	7463.00 0000	8003.00 0000	8138.00 0000	8069.00 0000	8378.00 0000

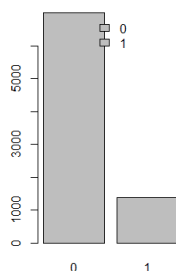
Como a variável objetivo é categórica (*match*), observou-se um problema de classificação binária, além disso, no conjunto de preditores temos variáveis numéricas (*age*, *age\_o*, e *int\_corr*) e variáveis categóricas (*goal*, *date*, *go\_out*, *length*, *met*, *like* e *prob*). Não foram considerados o *id* e o *partner* pois eles não agregam informação aos modelos.

Pôde-se ver pela tabela que nem todas as colunas têm a mesma contagem de elementos. Isso se deve aos valores nulos (N/A) nos dados, como por exemplo, em perguntas não respondidas. Por isso, foram substituídos os valores nulos: em *age* e *age\_o* pelo valor inteiro da média dessas variáveis, em *int\_corr* pelo *float* da média com duas casas decimais, e as variáveis categóricas são substituídas pela sua moda.

Utilizando a linguagem R, foram comparados os dados de cada variável ao *match*. Iremos omitir tabelas de frequência e probabilidade pela falta de espaço.

## Match

Essa é a nossa variável objetivo na qual sabemos ao final da conversa se há *match* (1) ou não *match* (0).

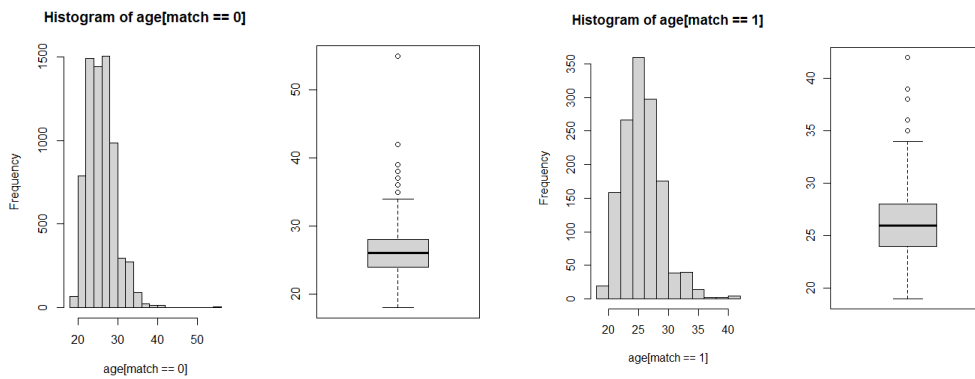


## Match

	0	1
Numérico	6998	1380
Probabilidade	0.8352829	0.1647171

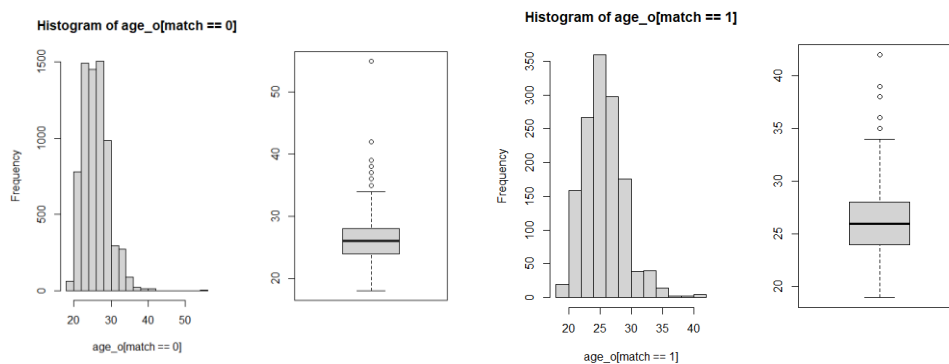
Foi possível observar que a probabilidade nessa amostra de as pessoas não darem *match* é muito maior do *match*.

## Idade



Em média, tanto as pessoas que não deram *match* quanto as pessoas que deram *match* tem 26 anos, e o desvio padrão dos dois conjuntos foi 3.6 e 3.3 respectivamente.

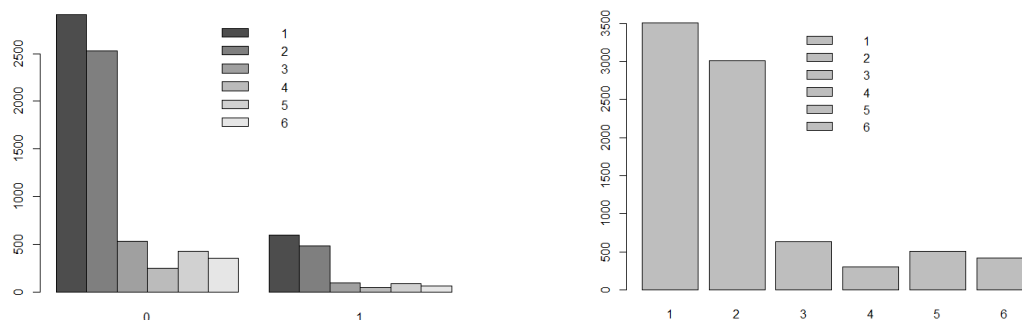
## Idade do par



Em média, tanto o par das pessoas que não deram *match* quanto os das pessoas que deram *match* tem 26 anos, e o desvio padrão dos dois conjuntos foi 3.6 e 3.3 respectivamente.

## Goal

Esse atributo mostra qual o objetivo principal ao participar neste evento. Pela tabela, o maior número de encontros provém das pessoas que têm o objetivo de passar uma noite divertida, seguido das pessoas que querem conhecer pessoas novas e o menor número de encontros se encontra nas pessoas que procuram um relacionamento sério.



No geral, para cada categoria existem mais pessoas que não deram *match*, logo, a categoria que a pessoa se encontra não afetou o *match*.

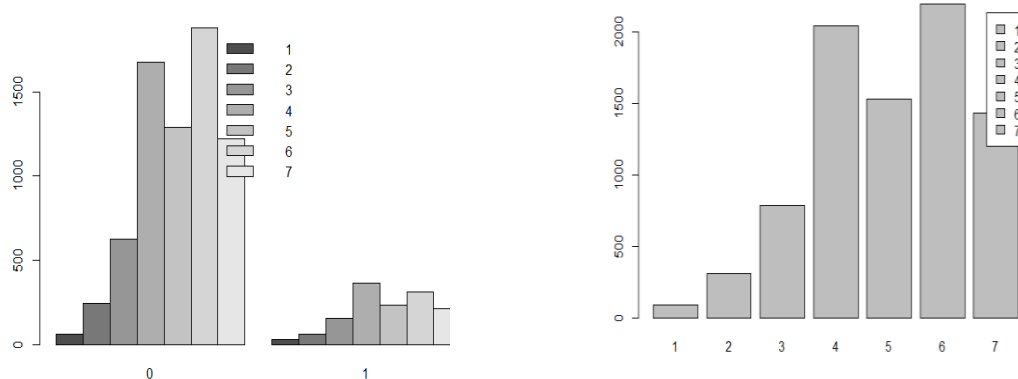
Observa-se a razão entre *matches*/ não *matches* em relação a cada motivação:

	1	2	3	4	5	6
<i>matches</i> / não <i>matches</i>	0.2057104	0.1914557	0.1883239	0.2040000	0.2000000	0.1736695

1 (passar uma noite divertida), 2 (conhecer pessoas novas), 3 (conseguir um encontro), 4 (procurar um relacionamento sério), 5 (dizer que consegui) e 6(outro).

Após fazer a razão entre os *matches* e os não *matches* foi obtida a tabela acima. Todas têm uma proporção de *match* parecida.

Date



Pelo gráfico de barras a maioria dos encontros se deram nas pessoas que saem para encontros 4 vezes por mês seguida das pessoas que saem para encontros várias vezes por ano o menor número se encontra nas pessoas que saem várias vezes por semana.

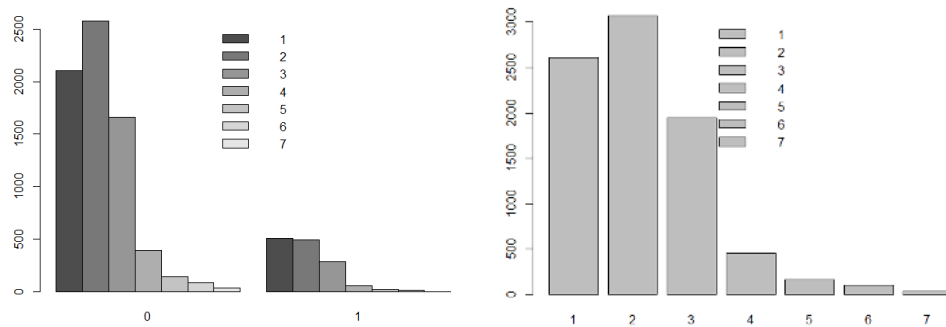
No geral podemos notar que para cada categoria existem mais pessoas que não deram *match*, logo a categoria que a pessoa se encontra não afeta o *match* ou não.

Nº Encontros	1	2	3	4	5	6	7
<i>matches</i> / não <i>matches</i>	0.4461538	0.2674897	0.2548077	0.2179104	0.1835786	0.1666667	0.1734861

Dentre as categorias, a categoria que deu mais *matches* foi a categoria 1, em que as pessoas vão a encontros várias vezes na semana, nesse caso, se diferencia bastante das outras, porém tem uma quantidade pequena na amostra.

Go out

A maioria dos encontros se dá nos indivíduos que saem duas vezes por semana, seguido dos indivíduos que saem várias vezes por semana, o menor número é nos indivíduos que quase nunca saem.

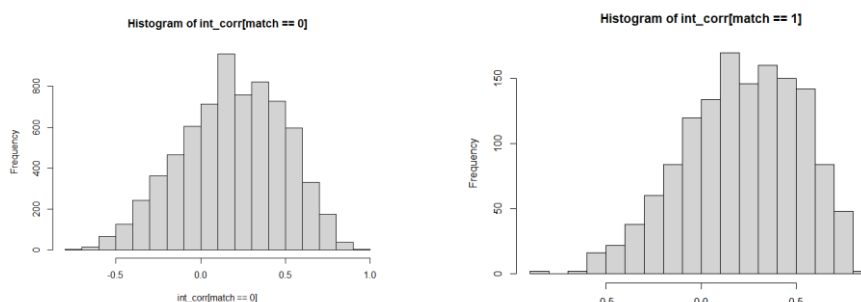


No geral, para cada categoria existe um número maior de rejeição em todos os casos, porém podemos observar que o maior número de não matches se encontra na segunda categoria, os que saem duas vezes por semana, e nos match primeira categoria, que saem várias vezes por semana.

	1	2	3	4	5	6	7
<i>matches / não matches</i>	0.2410841	0.19184466	0.17409639	0.14503817	0.13103448	0.15116279	0.02777778

Ao calcular a razão para cada categoria, vimos que a categoria que tem uma menor proporção de match dentre todas é a categoria 7, que se diferencia bastante das outras que possuem uma proporção bem parecida, apesar de estarem em menor quantidade.

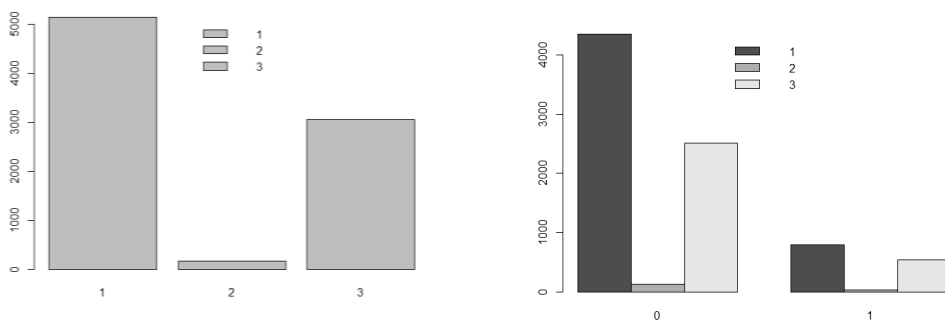
## Int\_corr



Os dois histogramas são aproximadamente simétricos, a média dos não matches é de uma correlação de 0.19 enquanto que a dos matches é de 0.22 tem correlações bem próximas, no geral também apresentam uma distribuição dos dados parecida. Mostrando que a correlação entre os ratings de interesses (desporto, museus, caminhadas, música, filmes, livros, etc.) do participante e do seu par não influencia tanto os matches.

## length

Pode-se ver que a maioria dos indivíduos achou a entrevista muito curta, seguido dos que acharam a entrevista com um tamanho adequado. Poucos indivíduos acharam longa.



para cada categoria há um maior número de rejeição. Observa-se a tabela da razão.

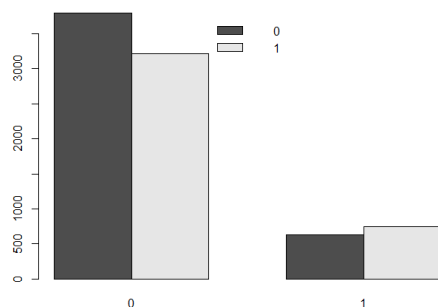
No geral ,

	1	2	3
<i>matches</i> / <i>não matches</i>	0.1828847	0.3111111	0.2158188

As pessoas que disseram que a entrevista era muito longa tiveram um maior índice de *match* dentre as outras duas categorias, porém ela representa uma quantidade muito pequena dos dados da amostra.

## Met

O número de encontros nos quais os pares que já se conheciam é parecido com os que não se conheciam.



Observamos que no grupo dos *matches* haviam mais pessoas que se conhecem do que as que não se conhecem, e dos que não deram match tem mais pessoas que não conheciam

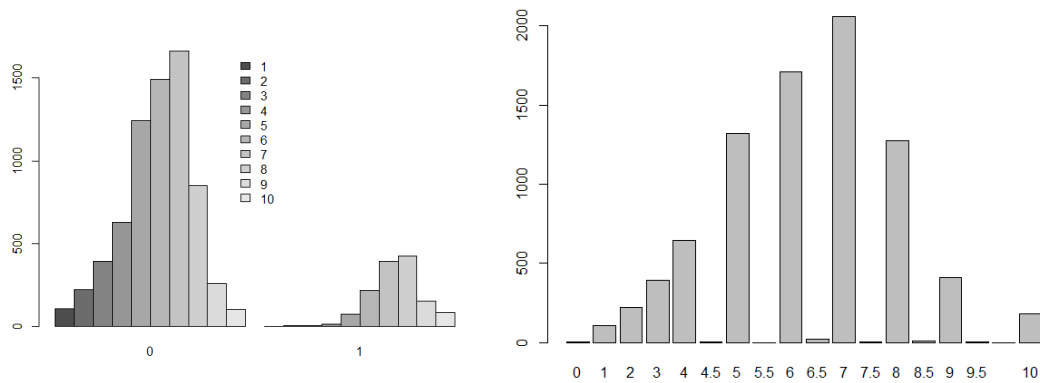
	0	1
<i>matches</i> / <i>não matches</i>	0.1673706	0.2323988

Ao fazermos a razão, observou-se que há mais *matches* quando as pessoas já se conhecem.

## Like

No gráfico do número de encontros para cada categoria, observa-se que os valores 0, 4.5, 5.5 etc. representam uma parcela muito pequena, por isso, para uma melhor análise gráfica e descritiva dos dados esses valores foram ignorados.





Na amostra existiam mais pessoas que gostaram do seu par em um nível 7, seguido de um nível 6. E no geral, também haviam mais não *matches* do que *matches*. Porém, observou-se que a distribuição dos não *matches* e dos *matches* teve uma leve diferença, com isso, observou-se a tabela da razão a seguir.

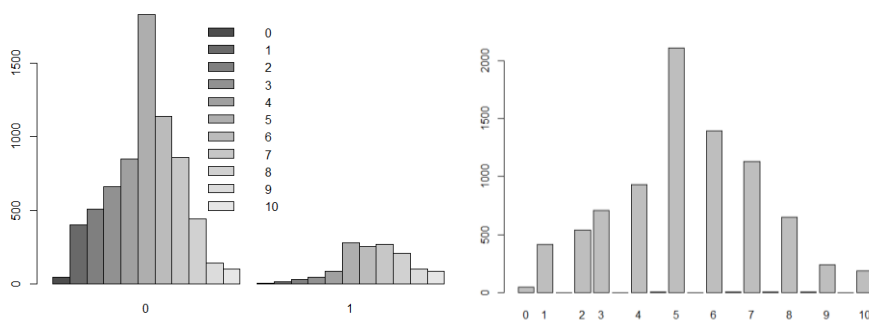
	1	2	3	4	5	6	7	8	9	10
<i>matches</i> /	0.0185	0.0136	0.0102	0.0238	0.0594	0.14390	0.2363	0.5005	0.590733	0.801980
<i>não matches</i>	1852	3636	0408	0952	3775	897	1990	8893	59	20

Pode-se ver que a proporção de *matches* é maior para os *likes* a partir do 6 até 10 e menor nos demais. Tem-se uma divisão clara dos dados nesse caso. Logo, a probabilidade de *match* é maior se a pessoa gosta da outra em uma escala de 6 a 10 e de não *match* numa escala de 0 a 5.

## Prob

Assim como antes, no gráfico do número de encontros para cada categoria pode-se ver que os valores 4.5, 5.5 etc. representam uma parcela muito pequena, por isso, para uma melhor análise gráfica e descritiva dos dados esses valores foram ignorados.

prob



Observou-se que a maior parte das pessoas disse que a probabilidade de o par ter gostado de si é de 5 e a menor é 0. E no geral, havia mais pessoas que não deram *match* do que *match* para cada

categoria. Mas, assim como na categoria anterior, a distribuição dos *matches* e dos não *matches* tem uma diferença.

	0	1	2	3	4	5	6	7	8	9	10
<i>match</i> <i>es</i> / não <i>match</i> <i>es</i>	0.0652 1739	0.0323 3831	0.0568 6275	0.0694 8640	0.0990 5660	0.1538 0405	0.2247 5856	0.3139 5349	0.4751 1312	0.6971 8310	0.8613 8614

Observou-se que conforme cresce a probabilidade de o par ter gostado do indivíduo, maior é a chance de dar *match*.

## Experiências e resultados

Foram implementados dois tipos de métodos de aprendizagem supervisionada: CART e Naive Bayes.

Para obtenção de uma estimativa de desempenho confiável, utilizou-se o método Holdout, em que se divide aleatoriamente o conjunto de dados em treino e teste. Utilizou-se 70% para treino e 30% para teste. Esse conjunto treino teste foi fixado (com o *random\_state=9*) para ser utilizado nos dois modelos.

### CART

Primeiramente fixou-se uma instância do modelo (*random\_state=9*). Ao fazer o modelo sem um *pre-pruning*, ou *post-pruning* a árvore retornada foi a imagem a seguir.



Portanto, é muito provável que ela esteja fazendo *overfitting* de dados, ou seja, ele não generaliza bem para exemplos não conhecidos. Nesse caso, para prevenir o sobreajustamento, optou-se pelo *pre-pruning*, ou pelo *post-pruning*.

O *pre-pruning* pode definir restrições para o tamanho da árvore, como:

1. Número mínimo de amostras para uma divisão de nós
2. Número mínimo de amostras para um nó de término
3. Profundidade máxima da árvore
4. Número máximo de nós de término
5. Número máximo de atributos a considerar para uma divisão

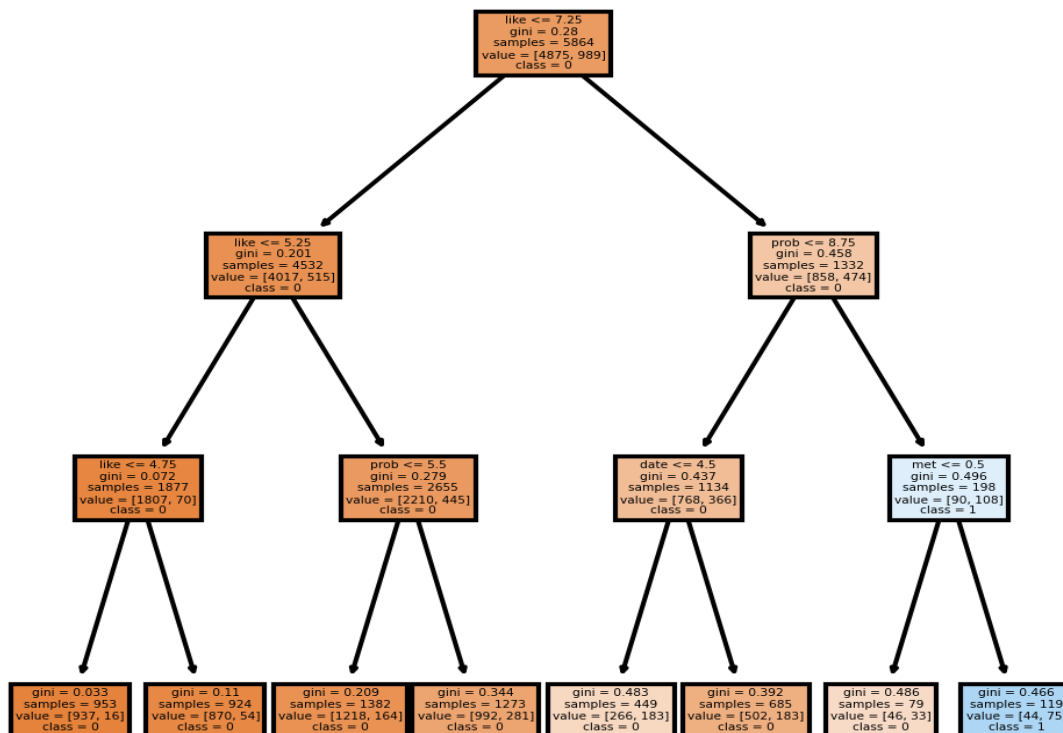
Já o post-pruning faz uma árvore de decisão com alto grau de profundidade e depois poda a árvore, removendo ramos com pouca qualidade de acordo com uma estimativa.

Ao analisar as opções pode-se perceber que as opções 1 e 2 não faziam sentido para o problema analisado, pois como a quantidade de valores para cada classe é desbalanceada seriam muito pequenas as regiões em que a classe *match* seria majoritária nos dados, logo não faz sentido definir um valor mínimo. Já o número máximo de nós de término não é adequado pois pode-se ter situações variadas que queira-se explorar, e excluiu-se o 5 pois iria selecionar aleatoriamente um número de variáveis a considerar, podendo excluir as mais significativas para o modelo.

Com relação ao *post-pruning*, o scikit-learn tem a opção *Post pruning decision trees with cost complexity pruning* que se decidiu por não explorar no momento.

Portanto, foi decidido uma restrição a profundidade da árvore para uma altura igual 3, pois a partir de um teste com alturas entre 1 e 100 no python, foi visto que essa altura deu uma melhor acurácia e melhor precisão (que voltará a ser tratado mais adiante), além de uma melhor visualização da árvore.

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272:  
_warn_prf(average, modifier, msg_start, len(result))  
melhor score: altura 3 melhor precision: altura 3 melhor recall: altura 18
```



Observou-se que na árvore gerada utilizando o algoritmo CART com *pre-pruning*, a primeira variável selecionada foi o *like*. Como visto na análise dos dados, ela é a que melhor divide os dados para o *match* ou *não match*, ou seja, apresenta uma menor “impureza de gini”, logo depois tem-se a *probabilidade*, que também resulta em folhas com um menor valor de Gini.

Viu-se que o algoritmo escolheu o *date* e o *met*, sendo que os separados por *met* apresentaram um valor de gini muito alto e resultou em folhas mais “incertas” sobre o seu resultado. O *date* também se destacou resultando em folhas com melhor valor de Gini do que o *met*. Como esperado, a maioria das folhas deram como predição a classe 0, isso foi devido a maioria dos encontros terem dado *não match*.

## Naive Bayes

Na análise dos dados as distribuições das variáveis não apresentam nenhum padrão específico mas são aproximadamente simétricas na maioria dos casos, logo, foi assumida uma distribuição normal para a população, ou seja, utilizou-se a probabilidade como gaussiana para o Naive Bayes.

Não foi necessária uma correção de laplace pois não obteve-se nenhuma probabilidade 0 nos dados.

## Avaliação

### Accuracy

A seguir tem-se uma tabela do valor do *accuracy* (1- (proporção das previsões incorretas)) , ou seja, quantos dos exemplos foram classificados corretamente.

	CART sem pruning	CART com pruning	Naive Bayes
Accuracy	76.17%	84,92%	83,17%

Tem que ter-se em consideração que nessa situação a acurácia pode ser enganosa, pois, caso considerado o *dummy model* que sempre prevê o rótulo mais frequente no conjunto de treinamento, no caso o *não match*, tem uma acurácia de 84,47%, ou seja tem-se que considerar outras métricas para não validar como modelo ótimo um modelo que falha em acertar os *matches*.

### Matriz de confusão, precision e recall

A matriz de confusão permite visualizar facilmente quantos exemplos foram classificados corretamente e erroneamente em cada classe, que ajuda a entender se o modelo está favorecendo uma classe em detrimento da outra.

		Classe prevista					
		CART sem pre-pruning		CART com pre-pruning		Naive Bayes	
		1	0	1	0	1	0
Classe verdadeira	1	130	261	34	357	95	296
	0	338	1785	22	2101	127	1996

A partir da matriz de confusão de cada um dos modelos, obteve-se as métricas *precision* e *recall*. Considerou-se como positivos os *matches* (1) e os negativos os *não matches*(0).

O *precision* dá a proporção de verdadeiros positivos dentre todos classificados como positivos e o *recall* é a proporção de previsão de positivos dentre aqueles que realmente eram positivos.

	CART sem pre-pruning	CART com pre-pruning	Naive Bayes
Precision (TruePositives/(TruePositives + FalsePositives))	27,78%	60,71%	42,80%
Recall (TruePositives/(TruePositives + FalseNegatives))	33,25%	8,70%	24,30%

Observou-se que para o *precision* o melhor modelo foi o CART com o pre- pruning, já para recall foi o CART sem o pre-pruning. Isso se deve ao fato de o CART c/ pre-pruning ter generalizado mais casos para os *não matches*, logo tem-se uma quantidade menor de falsos *matches* porém uma quantidade maior de falsos *não matches*. Já o CART s/ pre-pruning realizou uma análise mais específica, portanto chega em mais casos em que dá match.

O *Naive Bayes* tem um recall mais baixo em relação à acurácia . Isso se deve ao fato de o *Naive Bayes* considerar as probabilidades, e como há muito mais probabilidades de classificar categoria 0 na maioria dos casos, ele diminui a quantidade de previsões 1 erradas, o que aumenta o *precision* e aumenta previsões erradas de 0 diminuindo o recall.

A precisão pode ser usada em uma situação em que os Falsos Positivos são considerados mais prejudiciais que os Falsos Negativos, já o recall pode ser usado em casos em que os Falsos Negativos são considerados mais prejudiciais que os Falsos Positivos. No exemplo utilizado, dar *match* ou não, não se tem casos que seriam mais prejudiciais, então foi válido considerar a métrica F1-score que faz uma média harmônica entre precisão e recall.

	CART s/ pre-pruning	CART c/ pre-pruning	Naive Bayes
F1-score $\frac{2 \times \text{precisão} \times \text{recall}}{\text{precisão} + \text{recall}}$	30,27%	15,21%	31%

O F1-Score é uma maneira de observar somente uma métrica ao invés de duas em certas situações. O CART c/ pre-pruning é baixo, pois o recall é baixo.

## Limitações

No CART, para as variáveis categóricas se realiza uma divisão dos dados em subconjuntos, porém na árvore ele trata a divisão das variáveis categóricas como numéricas, ou seja, pelos pontos intermédios e não pelo subconjunto. Isso se deve pelo *scikit-learn* não oferecer suporte a variáveis categóricas.

Isso também acontece no *Naive Bayes* ao fazer um *Gaussian Naive Bayes*. Esse problema também poderia ser contornado ao usar *Categorical Naive Bayes* o que iria afetar as variáveis numéricas.

## Conclusão

Na análise exploratória dos dados, pôde-se ver que no geral os dados não dividem claramente os dados, e que em cada variável categórica a classe 0 tem uma quantidade maior do que 1. Porém em algumas situações tem-se uma separação mais clara dos dados, sendo estes escolhidos pelo CART para iniciar o modelo. O que faz com que os dois modelos sejam interpretáveis.

Foi possível concluir que o modelo gerado pelo *Naive Bayes* foi melhor dentre os demais, pois deu uma acurácia alta e apesar de não ter dado valores de precisão e recall altos, ainda se saiu melhor que os demais numa comparação geral dessas duas situações, ou seja, obteve um melhor F1-Score. Também foi possível observar que o CART com o *pruning* acabou por fazer *underfitting* dos dados, ou seja, gerou um modelo demasiado simples.

Também há outras hipóteses que poderiam ser consideradas como outras profundidades máximas para o CART, por exemplo a altura 18 que dava o melhor recall ou o *post-pruning* não considerado.

## Referências Bibliográficas

[https://pandas.pydata.org/docs/user\\_guide/index.html#user-guide](https://pandas.pydata.org/docs/user_guide/index.html#user-guide)

<https://medium.com/kunumi/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-em-machine-learning-classifica%C3%A7%C3%A3o-49340dcdb198>

<https://scikit-learn.org/stable/modules/tree.html>

[https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)

<https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html>

<https://www.vooo.pro/insights/um-tutorial-completo-sobre-a-modelagem-baseada-em-tree-arvore-do-zero-em-r-python/>