Nanodegree Engenheiro de Machine Learning

Projeto final

Marco Aurélio Moura Suriani
23 de abril de 2019

Previsão da necessidade de reforço escolar para alunos de matemática de acordo com características demográficas, sociais e acadêmicas

1. Definição

1.1. Visão geral do projeto

Como professor, carreira que exerço há sete anos, um dos meus grandes interesses é tentar prever o desempenho de alunos baseando-se em suas principais características. Com base neste tipo de previsão, é possível determinar, já desde o início das aulas, quais alunos precisarão de algum tipo de reforço, de modo a minimizar a quantidade de reprovações no fim do ano. Assim sendo, todo o trabalho de planejamento das atividades de reforço dependem de uma boa previsão de desempenho dos alunos.

Neste sentido, destaco o estudo de (Cortez e Silva, 2008), que faz parte de um esforço do governo de Portugal em melhorar os índices educacionais do país. Tal estudo buscou prever o desempenho (aprovado/reprovado) de estudantes na faixa de 15 a 22 anos em duas escolas a partir de características demográficas (como sexo e região), sociais (como escolaridade dos pais) e acadêmicas (como tempo semanal dedicado ao estudo e suporte externo), de modo a determinar se o aluno precisará de alguma forma de intervenção para melhorar seu desempenho acadêmico.

Já o estudo de (Kotsiantis; Pierrakeas e Pintelas, 2004) aplicou técnicas de Mineração de Dados para prever o desempenho de alunos universitários com base tanto em aspectos demográficos (como sexo e estado civil) quanto em avaliações passadas. A previsão foi realizada usando técnicas como Naive Bayes (que se mostrou a mais eficiente), Regressão Logística e k-Vizinhos mais Próximos (k-NN). Usando tais metodologias, os autores foram capazes de prever se um aluno seria aprovado ou reprovado com uma acurácia de 74%. Os resultados previram alunos que teriam fraco desempenho escolar na modalidade de ensino superior à distância, permitindo que os tutores tomem precauções e estejam mais bem preparados para lidar com casos do tipo.

O problema proposto neste trabalho é utilizar os mesmos dados usados no trabalho de (Cortez e Silva, 2008), que estão disponíveis no Conjunto de Dados de Desempenho de Estudantes do Repositório de Aprendizado de Máquinas da Universidade da Califórnia em Irvine, para tentar prever quais alunos precisarão de aulas de reforço na disciplina de Matemática com a maior antecedência possível. Sendo assim, a variável target será a necessidade de reforço, que é quantificável (o aluno precisa de reforço ou não) e mensurável (através de seu desempenho final).

Resumidamente, a primeira etapa do projeto foi escolher métricas que auxiliassem na escolha de bons modelos de previsão. Em seguida, os dados brutos foram pré-processados e usados por um conjunto de técnicas e algoritmos de Aprendizado de Máquinas paa classificação. A próxima etapa foi selecionar os melhores atributos e os melhores algoritmos para gerar um modelo final de previsões. O modelo final gerado obteve uma acurácia 9% superior à de um previsor que classifica todos alunos como aprovados, bem como um F1-Score (β = 0,5) 48% superior ao de um previsor que classifica todos so alunos como reprovados. Tais resultados podem ser considerados satisfatórios.

1.2. Descrição do problema

O Conjunto de Dados de Desempenho de Estudantes contém características demográficas, sociais e acadêmicas de um grupo de estudantes, bem como suas respectivas notas parciais e finais em português e em matemática. Dentre as características dos alunos, estão sexo, idade, educação dos pais, qualidade das relações familiares, tempo de deslocamento entre casa e escola, frequência com que sai com os amigos, consumo de bebida alcóolica e acesso á internet.

O problema a ser resolvido é determinar se um aluno precisará de reforço em matemática baseado apenas nos atributos conhecidos no início do aluno letivo, ou seja, sem levar em consideração as notas parciais obtidas antes da nota final, e nem a quantidade de faltas. Um aluno que precisa de reforço é aquele do qual se espera que não seja capaz de atingir nota mínima para aprovação no fim do ano letivo.

A diferença entre o presente projeto e os que foram citados anteriormente é que tais trabalhos se limitaram a medir a acurácia das previsões de seus respectivos modelos. Tal métrica costuma ser de pouca utilidade paa a análise de conjuntos desbalanceados de dados como os de desempenho acadêmico (geralmente, a quantidade de alunos aprovados é muito maior do que a de reprovados). Além disso, tais trabalhos construíram seus modelos de previsão de acordo com características demográficas, sociais e acadêmicas dos estudantes, e concluíram que as notas dos alunos ao longo do ano são o melhor atributo para prever o desempenho final. Entretanto, o uso de notas parciais no modelo de previsão é problemático pois tais valores só ficam disponíveis no decorrer do ano, impedindo um planejamento com antecedência suficiente para evitar as reprovações.

Por estes motivos, este projeto tem como objetivo desenvolver um modelo de previsões com auxílio de métricas mais apropriadas para as características do conjunto de dados e do problema. Além disso, as métricas escolhidas levarão em conta que as previsões visam saber quais alunos precisam de reforço e não quais alunos serão reprovados. Esta alteração no enquadramento será fundamental na escolha da métrica para avaliar os modelos. Por fim, os modelos usarão apenas atributos que podem estar disponíveis no início das aulas, abrindo mão de variáveis como notas parciais e notas, de modo que possam fazer previsões o mais cedo possível.

A solução para este problema será um modelo de previsão da necessidade de cada aluno de receber aulas de reforço a partir de um modelo de Aprendizado de Máquinas. Tal modelo deve identificar o máximo possível de alunos que precisam de reforço, enquanto evita classificar erroneamente os alunos que não precisam. Por fim, a solução pode ser mensurada através da quantidade de previsões de necessidade de reforço e pode ser replicada quantas vezes for necessário a partir do modelo que será desenvolvido.

Etapas do projeto:

1. **Escolha de métricas** que possam ser úteis, tais como acurácia, precisão, revocação e F1-score (inclusive a escolha de seu parâmetro β). As métricas devem se adequar às características do conjunto de dados e ao problema proposto;
2. **Pré-processamento** dos atributos para que possam ser utilizados nos modelos propostos. Alguns atributos são binários, devendo ser codificados em 0/1, enquanto outros são nominais, devendo ser codificados usando uma técnica como one-hot;
3. **Aplicação de algoritmos de Aprendizado de Máquinas para classificação** para prever a necessidade de reforço escolar aos estudantes. Serão usados a Regressão Logística, o Naive Bayes, as Árvores de Decisão, as Máquinas de Vetores de Suporte, o Classificador AdaBoost e os k-Vizinhos mais Próximos (k-NN). Os modelos serão treinados usando 80% dos dados e testados usando os 20% restantes (validação cruzada). Os resultados serão comparados entre si através das métricas escolhidas. Além disto, o desempenho dos modelos será comparado ao desempenho de um preditor ingênuo, que classifica todos os estudantes como pertencentes à categoria com maior frequência.
4. Será realizada uma **seleção de atributos** para tentar diminuir a quantidade de atributos e melhorar a qualidade das previsões. Em seguida, uma nova rodada de testes com os melhores algoritmos da etapa anterior será realizada com os atributos selecionados.
5. O melhor modelo será usado para apontar a solução para o problema.

1.3. Métricas

Primeiramente, conforme observado a seção anterior, o trabalho de (Cortez e Silva, 2008) leva em consideração apenas a acurácia para avaliação dos modelos, mas para o propósito deste trabalho, apenas tal métrica não será o bastante. Também deverão ser levadas em conta métricas como a precisão e a revocação (recall) para evitar que alunos sem reforço acabem reprovados e que alunos que acabariam sendo aprovados sejam submetidos desnecessariamente a aulas de reforço. Por fim, também será usado o F score, que é a média harmônica entre precisão e revocação.

Todos estes valores podem ser calculados a partir de **Matriz de Confusão**. Tal matriz relaciona o target verdadeiro de cada instância com seu valor previsto. Desta forma, uma instância pode ser um Verdadeiro Positivo (se classificada corretamente como 1) ou Verdadeiro Negativo (se classificada corretamente como 0), bem como um Falso Positivo (se classificada equivocadamente como 1) ou Falso Negativo (se classificada equivocadamente como 0). (GeeksforGeeks, Confusion Matrix in Machine Learning) Resumidamente:

Matriz de Confusão	Classificado como 0	Classificado como 1
Valor Real 0	Verdadeiro Negativo **(VN)**	Falso Positivo **(FP)**
Valor Real 1	Falso Negativo **(FN)**	Verdadeiro Positivo **(VP)**

As métricas usadas neste trabalho são:

1. **Acurácia (accuracy):** mede com que frequência o classificador faz a predição correta. É a proporção entre o número de predições corretas e o número total de predições (o número de registros testados), ou seja:

$$Acurácia = \frac{VN + VP}{Total}$$

2. **Precisão (precision):** informa qual a proporção de classificações positivas eram realmente positivas. É a proporção entre o número de verdadeiros positivos o total de classificados como positivos, ou seja:

$$Precisão = \frac{VP}{VP + FP}$$

3. **Revocação (recall):** informa qual a proporção de istâncias positivas que foram corretamente classificadas como positias. É a proporção entre o número de verdadeiros positivos e o total de instâncias positivas, ou seja:

$$Revocação = \frac{VP}{VP + FN}$$

4. **F_β score:** média harmônica entre precisão e revocação. Possui um parâmetro β que permite dar mais peso à precisão ($0 < \beta < 1$) ou à revocação ($\beta > 1$), ou seja:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\left(\beta^2 \cdot precision\right) + recall}$$

Em termos do presente problema, pode-se afirmar que a precisão mede a eficiência dos gastos (pois diminui quando alunos que seriam aprovados de qualquer forma são classificados como candidatos a reforço) e que a revocação mede a eficácia da previsão (pois diminui quando alunos que seriam reprovados não são classificados como candidats a reforço).

Como a meta deste modelo é auxiliar na previsão de alunos que precisam de reforço (ao invés de simplesmente fazer o máximo possível de previsões de desempenho), foi escolhido um β = 0,5 que valoriza a precisão e, portanto, a eficiência dos gastos. Desta forma, como o F score a leva em conta precisão e revocação, as métricas escolhias são:

1. Acurácia
2. $F_{0,5}\ score$

Por fim, o tempo de processamento para treino e previsão também será levado em conta.

---

2. Análise

2.1. Exploração dos dados

O conjunto de dados do repositório contém 33 variáveis para alunos de Matemática (395 instâncias, **sem dados ausentes**), sendo elas:
1- school - escola (binária: 'GP' - Gabriel Pereira ou 'MS' - Mousinho da Silveira),
2- sex - sexo do estudante (binária: 'F' - feminino ou 'M' - masculino),
3- age - idade (numérica: 15 a 22),
4- address - tipo de residência (binária: 'U' - urbana ou 'R' - rural),
5- famsize - tamanho da família (binária: 'LE3' - até 3 ou 'GT3' - mais que 3),
6- Pstatus - cohabitação com os pais (binária: 'T' - moram juntos ou 'A' - separados),
7- Medu - educação da mãe (categórica ordinal: 0 - nenhuma, 1 - até 4º ano, 2 - até 9º ano, 3 - ensino médio ou 4 - ensino superior),
8- Fedu - eduação do pai (categórica ordinal: idem anterior),
9- Mjob - emprego da mãe (nominal: 'teacher', 'health', 'services', 'at_home' ou 'other'),
10- Fjob - emprego do pai (nominal: idem anterior),
11- reason - motivo de escolha da escola (nominal: 'home' - próximo de casa, 'reputation' - reputação da escola, 'course' - preferência pelo curso, ou 'other'),
12- guardian - guarda do aluno (nominal: 'mother', 'father' ou 'other'),
13- traveltime - tempo de percurso até a escola (categórica ordinal: 1 - 1 hora),
14- studytime - tempo de estudo semanal (categórica ordinal: 1 - 10 horas),
15- failures - reprovações passadas (numérica: n se 1<=n<3, caso contrário 4),
16- schoolsup - suporte extra (binária: 'yes' ou 'no'),
17- famsup - suporte familiar (binária: 'yes' ou 'no'),
18- paid - aulas particulares pagas (binária: 'yes' ou 'no'),
19- activities - atividades extra-curriculares (binária: 'yes' ou 'no'),
20- nursery - cursou a pré-escola (binária: 'yes' ou 'no'),
21- higher - deseja cursar ensino superior (binária: 'yes' ou 'no'),
22- internet - acesso à Internet (binária: 'yes' ou 'no'),
23- romantic - em relacionamento amoroso (binária: 'yes' ou 'no'),
24- famrel - qualidade das relações familiares (categórica ordinal: de 1 - muito ruim a 5 - excelente),
25- freetime - tempo livre após a escola (categórica ordinal: de 1 - muito baixo a 5 - muito alto),
26- goout - sai com amigos (categórica ordinal: de 1 - muito baixo a 5 - muito alto),
27- Dalc - consumo de álcool durante a semana (categórica ordinal: de 1 - muito baixo a 5 - muito alto),
28- Walc - consumo de álcool durante o fim-de-semana (categórica ordinal: de 1 - muito baixo a 5 - muito alto),
29- health - estado de saúde (categórica ordinal: de 1 - muito baixo a 5 - muito alto),
30- absences - faltas à escola (numérica: de 0 a 93),
31- G1 - nota no primeiro período (numérica: de 0 a 20),
32- G2 - nota no segundo período (numérica: de 0 a 20),
33- G3 - nota final (numérica: de 0 a 20, target).

Destas, as 29 primeiras são entradas e as 3 últimas são saídas. A 30ª variável (absenses ou faltas)não tem utilidade alguma pois ela só é conhecida quando as notas estão fechadas (então não pode ser usada para previsões) e nem tem relevância prática para que necessitemos prevê-la. Assim sendo, não é atributo nem alvo.

As 5 primeiras instâncias deste conjunto de dados podem ser vistas na figura a seguir:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	
1	GP	F	17	U	GT3	T	1	1	at_home	other	
2	GP	F	15	U	LE3	T	1	1	at_home	other	
3	GP	F	15	U	GT3	T	4	2	health	services	
4	GP	F	16	U	GT3	T	3	3	other	other	...

	reason	guardian	traveltime	studytime	failures	schoolsup	famsup	paid	
0	course	mother	2	2	0	yes	no	no	
1	course	father	1	2	0	no	yes	no	
2	other	mother	1	2	3	yes	no	yes	
3	home	mother	1	3	0	no	yes	yes	
4	home	father	1	2	0	no	yes	yes	...

	activities	nursery	higher	internet	romantic	famrel	freetime	goout	
0	no	yes	yes	no	no	4	3	4	
1	no	no	yes	yes	no	5	3	3	
2	no	yes	yes	yes	no	4	3	2	
3	yes	yes	yes	yes	yes	3	2	2	
4	no	yes	yes	no	no	4	3	2	...

	Dalc	Walc	health	absences	G1	G2	G3
0	1	1	3	6	5	6	6
1	1	1	3	4	5	5	6
2	2	3	3	10	7	8	10
3	1	1	5	2	15	14	15
4	1	2	5	4	6	10	10

Das 33 variáveis, pode-se observar que:

- 6 são numéricas discretas;
- 10 são categóricas ordinais (é importante ressaltar que tais variáveis são categóricas pois expressam avaliações pessoais, ao invés de intensidades ou contagens; entretanto, por serem ordinais, elas podem ser expressas através de números discretos; no caso deste *dataset*, todas elas vieram expressas como números, poupando trabalho de pré-processamento de dados);
- 4 são categóricas nominais (por não haver hierarquia entre as categorias, não podem ser expressas através de números discretos; a solução mais comum é usar o método *one-hot encondig*);
- 13 são categóricas binárias (que podem ser expressas através de 0/1).

Algumas estatísticas sobre as variáveis numéricas e sobre as variáveis categóricas ordinais (que já estão em formato numérico), podem ser vistas na tabela a seguir. Como as escalas destas variáveis são muito diferentes (idade vai de 15 a 22 com média 16,7 enquanto falhas vai de 0 a 3 com média 0,33), então seria uma boa ideia trazer todas para a mesma escala.

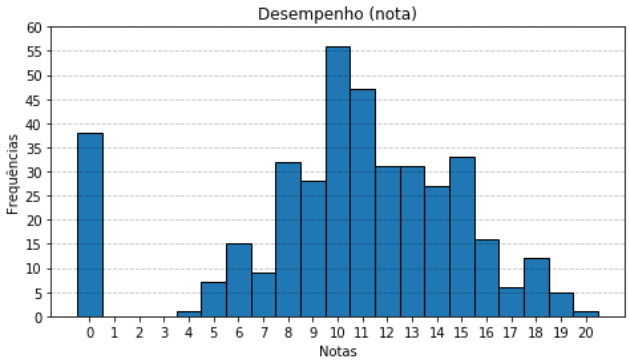
	age	Medu	Fedu	traveltime	studytime	failures	\
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	
std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	

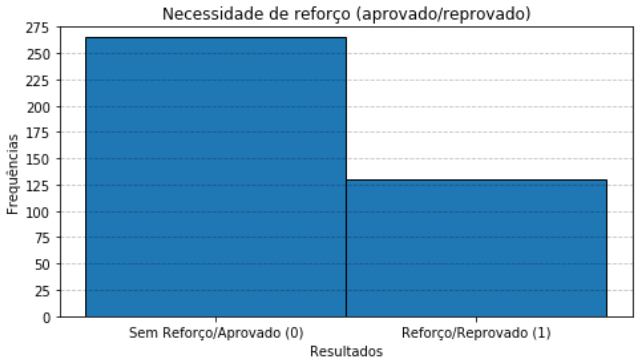
	famrel	freetime	goout	Dalc	Walc	health	\
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	
mean	3.944304	3.235443	3.108861	1.481013	2.291139	3.554430	
std	0.896659	0.998862	1.113278	0.890741	1.287897	1.390303	
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	
25%	4.000000	3.000000	2.000000	1.000000	1.000000	3.000000	
50%	4.000000	3.000000	3.000000	1.000000	2.000000	4.000000	
75%	5.000000	4.000000	4.000000	2.000000	3.000000	5.000000	
max	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	

	absences	G1	G2	G3
count	395.000000	395.000000	395.000000	395.000000
mean	5.708861	10.908861	10.713924	10.415190
std	8.003096	3.319195	3.761505	4.581443
min	0.000000	3.000000	0.000000	0.000000
25%	0.000000	8.000000	9.000000	8.000000
50%	4.000000	11.000000	11.000000	11.000000
75%	8.000000	13.000000	13.000000	14.000000
max	75.000000	19.000000	19.000000	20.000000

2.2. Visualização exploratória

A seguir, estão dois gráficos que ajudam a compreender o problema. O primeiro é um histograma com as notas dos alunos. Já o segundo exibe um um gráfico de barras com os totais de alunos aprovados e reprovados do daaset.





Algumas conclusões pdem ser tiradas da gravuras:

- Excluindo as notas 0, a distribuição de G3 é aproximadamente normal;
- Observam-se muitas notas 0 (38 ocorrências, 3ª nota mais frequente), provavelmente desistentes - entretanto, não se sabem as causas das desistências: mudança de escola, mudança de cidade, desistência de ensino, problema de saúde, falecimento etc;
- A nota mínima para aprovação (10) é a mais frequente (56 ocorrências), seguida pela nota 11 (47 ocorrências) - provavelmente muitos desses alunos foram "empurrados" pelo professor ou passaram no limite, sendo tais possibilidades possíveis de ser exploradas em outros trabalhos;
- A quantidade de alunos aprovados é o dobro da quantidade de alunos reprovados - tal desbalanceamente torna a acurácia uma medida praticamente inútil, pois pode-se obter uma acurácia de 0,67 simplesmente classificando todos os alunos como "Sem Reforço".

2.3. Algoritmos e técnicas

Os algoritmos escolhidos foram:

Método 1: Regressão Logística

1. A Regressão Logística, assim como as Máquinas de Suporte Vetorial, é um modelo linear. Desta forma, é um modelo rápido para treinar e para prever, mas pode ter dificuldades em fazer boas previsões com fronteiras não-lineares.
2. Este modelo consegue trabalhar bem com datasets muito grandes e com uma grande quantidade de atributos, apesar de não ser ideal para datasets menores. (Mueller e Guido, 2016)
3. Como é um modelo rápido e que apresenta bons resultados para uma quantidade grande de atributos, talvez o modelo `LogisticRegression` consiga bons resultados. Serão usados os parâmetros default.

Método 2: Naïve Bayes

1. O modelo é eficiente, ou seja, ele treina e classifica muito rapidamente. Este modelo é eficiente porque aprende os parâmetros analisando cada atributo individualmente. Inclusive, ele é chamado de naive (ingênuo) justamente por presseupor que os atributos são independentes entre si. Porém, o modelo pode apresentar queda na acurácia caso os atributos sejam altamente correlacionados entre si.
2. Possui 3 versões: MultinomialNB (próprio para dados esparsos, como contagem de palavras em texto), BernoulliNB (próprio para dados binários) e GaussianNB (próprio para dados contínuos). (Mueller e Guido, 2016)
3. Neste caso, o algoritmo mais adequado é o `GaussianNB` . Serão usados os parâmetros default.

Método 3: AdaBoost

1. O modelo é rápido, fácil de programar, necessita de poucos parâmetros de ajuste, pode ser combinado com qualquer algoritmo de aprendizado que faz o papel de classificador fraco (o caso mais comum é a Árvore de Decisão) e comprovadamente eficiente. Por outro ladeo, seu desempenho depende da escolha do classificador fraco. Pode levar a sobreajuste caso o classificador fraco seja muito complexo.
2. Pode modelar dados textuais, numéricos e discretos, bem como imagens. Por outro lado, a experiência tem mostrado que o Adaboost é particularmente sucetível a ruido uniforme. (Schapire, A Boosting Tutorial)
3. O algoritmo `AdaBoostClassifier` é bastante eficiente e tem bom desempenho independentemente do tipo de dados. Como não é esperado que haja ruido nos dados, este modelo deve funcionar bem. Serão usados os parâmetros default.

Outros métodos

Os métodos a seguir também foram aplicados na tentativa de se estabelecer um modelo, mas não se espera que alcancem bom desempenho. Eles estão sendo aplicados apenas com o objetivo de ter suas métricas comparadas com as métricas dos métodos anteriores para fins de comparação.

- **Método 4: k-Vizinhos mais Próximos (kNN).** Como há muitos atributos, dificilmente tal método trará bons resultados em termos de acurácia e F0,5-score. Serão usados 3 vizinhos como parâmetro.
- **Método 5: Classificação via Máquinas de Suporte de Vetores (SVC).** Como há muitos atributos, dificilmente tal método trará bons resultados em termos de tempo de processamento. Serão usados como parâmetros kernel `linear` e C = 100.
- **Método 6: Árvores de Decisão.** Não espero que consiga performar melhor do que o AdaBoost. Serão usados os parâmetros default, exceto min_samples_leaf = 3.

2.4. Benchmark

O trabalho de (Cortez e Silva, 2008), que é a fonte dos dados usados neste trabalho, determinou três configurações de entrada para os modelos de previsão: (A) aquele que levava em consideração as notas parciais G1 e G2 para prever se o aluno obteria uma G3 abaixo de 10 (reprovado) ou a partir de 10 (aprovado), (B) aquele que levava em consideração apenas G1 e (C) aquele que levava em consderação apenas os demais atributos (sem considerar as notas parciais). *O presente trabalho usará os resultados da configuração (C) como base para comparações.* Os resultados podem ser observados na tabela abaixo.

Binary classification results
(PCC values, in %; underline – best model; **bold** – best within the input setup)

Input Setup	Mathematics				
	NV	NN	SVM	DT	RF
A	**91.9**$^{\dagger}_{\pm 0.0}$	88.3$_{\pm 0.7}$	86.3$_{\pm 0.6}$	90.7$_{\pm 0.3}$	91.2$_{\pm 0.2}$
B	**83.8**$^{\dagger}_{\pm 0.0}$	81.3$_{\pm 0.5}$	80.5$_{\pm 0.5}$	83.1$_{\pm 0.5}$	83.0$_{\pm 0.4}$
C	67.1$_{\pm 0.0}$	66.3$_{\pm 1.0}$	**70.6**$^{*}_{\pm 0.4}$	65.3$_{\pm 0.8}$	70.5$_{\pm 0.5}$

† – statistical significance under pairwise comparisons with other methods.
∗ – statistical significance under a pairwise comparison with NV.

Para cada configuração de entrada, a previsão foi realizada através das seguintes técnicas de Mineração de Dados: Árvores de Decisão, Florestas Aleatórias, Redes Neurais e Máquinas de Vetores de Suporte. A avaliação dos modelos se deu por sua acurácia (#previsões corretas/#previsões). Os resultados mostram que as acurácias obtidas dependem menos do tipo de modelo do que da configuração de entrada, uma vez que a configuração (A), que leva em consideração as notas dos alunos ao longo do ano, sempre obteve melhor acurácia do que a configuração (C), que não leva em conta o desempenho passado. Mais especificamente, observaram-se na configuração (C) acurácias entre 65,3% (Árvore de Decisão) e 70,6% (Máquinas de Vetores de Suporte). Vale ressaltar que na configuração (C), um preditor ingênuo que classifica todos os estudantes como aprovados obtém acurácia de 67,1%. Além disso, nas configurações (A) e (B), nenhum modelo superou o previsor ingênuo.

---

3. Metodologia

3.1. Pré-processamento de dados

A etapa de pré-processamento contém os seguintes passos:

1. Criar uma cópia do dataset original que possa ser alterada livremente;
2. Alterar a escala **(Rescale data)** das variáveias numéricas de entrada e das variáveis categóricas ordinais (que já estão expressas através de números discretos) para a escala [0, 1]. Desta forma, o menor valor de cada variável se torna 0, o maior se torna 1, e os demais valores são calculados proporcionalmente (GeeksforGeeks, Data Preprocessing for Machine learning in Python);

In [ ]:
```
# Variáveis numéricas e categóricas ordinárias de entrada (Rescale data)
for var in var_num_disc+var_cat_ordn:
    if var in var_input:
            vmin = np.min(mydataset1[[var]])
            d = np.max(mydataset1[[var]]) - vmin
            mydataset1[[var]] = (mydataset1[[var]] - vmin) / d
```

3. Converter em números binários **(Binarize data)** as variáveis categóricas binárias. (GeeksforGeeks, Data Preprocessing for Machine learning in Python) Uma das categorias será 0 e a outra será 1, sendo que tal escolha é arbitrária e não influencia o resultado final;

In [ ]:
```
# Variáveis Binárias de entrada (Binarize data)
mydataset1.school = mydataset1.school.apply(lambda x: 1 if x == 'GP' else 0)
mydataset1.sex = mydataset1.sex.apply(lambda x: 1 if x == 'M' else 0)
mydataset1.address = mydataset1.address.apply(lambda x: 1 if x == 'U' else 0)
```

4. Codificar através do método one-hot **(One-hot encoding)** as variáveis categóricas nominais. Tal método cria variáveis *dummy* do tipo binárias para cada categoria da variável original. (Kaggle, Using Categorical Data with One Hot Encoding) Em seguida, as variáveis originais são apagadas.

In [ ]:
```
# Variáveis Nominais (One-hot encoding)
dfDummies = pd.get_dummies(mydataset1['Mjob'], prefix='Mjob')
mydataset1 = pd.concat([mydataset1, dfDummies], axis=1)

dfDummies = pd.get_dummies(mydataset1['Fjob'], prefix='Fjob')
mydataset1 = pd.concat([mydataset1, dfDummies], axis=1)

mydataset1.drop(['Mjob', 'Fjob', 'reason', 'guardian'], 1, inplace=True)
```

5. Em seguida, são criadas as variáveis de atributos (*features*) e de alvos (*targets*):

   - ***Atributos (X):*** Variáveis pré-processadas de entrada (excluem-se as saídas 'absences' e as notas 'G1 a 'G3');
   - ***Alvo numérico (Y):*** Variável de saída (nota G3) expressa numericamente;
   - ***Alvo binário (Yb):*** Variável de saída transformada em binário:
     - **0:** Aluno não precisa de reforço (nota G3 >= 10)
     - **1:** Aluno **precisa** de reforço (nota G3 < 10)

In [ ]:
```
# Entradas e Saídas
X = mydataset1.drop(var_output, 1)
Y = mydataset1[['G3']]
Yb = Y.G3.apply(lambda x: 0 if x>=10 else 1)
```

Os códigos completos estão no arquivo `code`. A seguir, uma lista com todos os atributos da variável X:

```
01) school              02) sex                 03) age                 04) address
05) famsize             06) Pstatus             07) Medu                08) Fedu
09) traveltime          10) studytime           11) failures            12) schoolsup
13) famsup              14) paid                15) activities          16) nursery
17) higher              18) internet            19) romantic            20) famrel
21) freetime            22) goout               23) Dalc                24) Walc
25) health              26) Mjob_at_home        27) Mjob_health         28) Mjob_other
29) Mjob_services       30) Mjob_teacher        31) Fjob_at_home        32) Fjob_health
33) Fjob_other          34) Fjob_services       35) Fjob_teacher        36) reason_course
37) reason_home         38) reason_other        39) reason_reputation   40) guardian_father
41) guardian_mother     42) guardian_other
```

3.2. Implementação

Nesta etapa, os seis algoritmos de modelagem ciados anteriormente foram implementados, testados, medidos e comparados. A medição envolveu a acurácia, o F0,5-score e o tempo de processamento. A comparação envolveu modelos ingênuos que aprovavam ou reprovavam todos os alunos. Por fim, selecionou-se os três melhores modelos para a próxima etapa.

Etapa 3.2.a: Funções de Suporte

Codificação de funções que serão usadas ao longo do trabalho:

1. Métricas: Função **metricas_binario(y_test, y_pred, verbose)**:

   - Compara **y_test** ao **y_pred**, determinando:
     - Matriz de Confusão
     - Acurácia
     - Precisão, Revocação e F_β-score com β = 0,5 (ou $F_{0,5}$)
   - Retorna vetor contendo:
     - Acurácia
     - $F_{0,5}$
   - Se **verbose**, então imprime os resultados obtidos.

2. Pipeline: Função **model_pipeline(X, y, model)**:

   - Recebe um dataset como entrada **(X, y)** e o divide em subconjuntos de treino e de testes, com 20% para testes;
   - Recebe um objeto **model** do modelo de previsão como entrada e usa seu método **.fit** com o subconjunto de treino;
   - Usa o método **.predict** do modelo para determinar os valores previstos de saída **(y_pred)**;
   - Chama a função **metricas_binario( )** para avaliar as previsões y_pred em comparação ao alvo y_test, e retorna acurácia e F0,5-score.

In [ ]:

```python
#Funções de Suporte

def metricas_binario(y_test, y_pred, verbose=True):
    cm = confusion_matrix(y_test, y_pred)
    acc = accuracy_score(y_test, y_pred)
    scores = precision_recall_fscore_support(y_test, y_pred, beta=0.5, warn_for=())

    if verbose:
        print('Matriz de Confusão\n', cm[0], '\n', cm[1], '\n')

        print('Acurácia  = {:.3f}'.format(acc))
        print('Precisão  = {:.3f}'.format(scores[0][1]))
        print('Revocação = {:.3f}'.format(scores[1][1]))
        print('F0,5 scr  = {:.3f}'.format(scores[2][1]))

    return [acc, scores[2][1]]

def model_pipeline(X, Y, model, verbose=False):
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    metricas = metricas_binario(y_test, y_pred, verbose)

    return metricas
```

Etapa 3.2.b: Classes com previsores ingênuos

1. Previsor ingênuo (prevê que todos serão aprovados): Cria a classe Previsor_ingenuo_aprova com os seguintes métodos:
   - fit(X, y): não faz nada (precisa existir por causa da função model_pipeline( ));
   - predict(x): prevê 0 (aprovado, não precisa de reforço) para toda instância em x.

2. Previsor ingênuo (prevê que todos serão reprovados): Cria a classe Previsor_ingenuo_reprova com os seguintes métodos:
   - fit(X, y): não faz nada (precisa existir por causa da função model_pipeline( ));
   - predict(x): prevê 1 (reprovado, precisa de reforço) para toda instância em x.

In [ ]:

```python
# CLASSE Previsor Ingênuo (prevê apenas aprovados)
class Previsor_ingenuo_aprova:
    def fit(self, X, y):
        pass

    def predict(self, X):
        return pd.DataFrame([0 for x in range(X.shape[0])]).iloc[:,0]

# CLASSE Previsor Ingênuo (prevê apenas reprovados)
class Previsor_ingenuo_reprova:
    def fit(self, X, y):
        pass

    def predict(self, X):
        return pd.DataFrame([1 for x in range(X.shape[0])]).iloc[:,0]
```

Etapa 3.2.c: Comparação acurácia e F0,5 para diferentes modelos de classificação

Esta etapa tem como objetivo determinar e comparar a acurácia e o F0,5 dos seis algoritmos de classificação escolhidos (mais os dois previsores ingênuos) usando como alvo o desempenho final dos estudantaes (aprovado=0/reprovado=1). Os passos desta etapa são:

- Criar um Data Frame chamado **metrics** para armazenar as métricas (acurácia e F0,5-score) de cada técnica, bem como seu tempo médio para treino e previsão;
- Aplicar diversos modelos e gravar seus resultados em **metrics**:
  - Calcula a média da acurácia e do F0,5-score para 1000 previsões de cada técnica;
  - Mede o tempo de treino e previsão;
  - Grava e imprime as médias;
  - Calcula um modelo usando **verbose** para imprimir matriz de confusão, precisão e revocação, como exemplo (apenas para algumas das técnicas).
- Exibir os resultados em uma tabela e em um gráfico;
- Analisar os resultados e tirar conclusões.

A seguir, o código para o Data Frame **metrics**, seguido da programação para a `Regressão Logística` , bem como os respectivos resultados que são impressos para o usuário. A programação dos demais algoritmos está no arquivo `code` .

In [ ]:
```
# Data Frame para armazenar as métricas
metrics = pd.DataFrame(columns = ['Acurácia', 'F0,5', 'Tempo'])
```

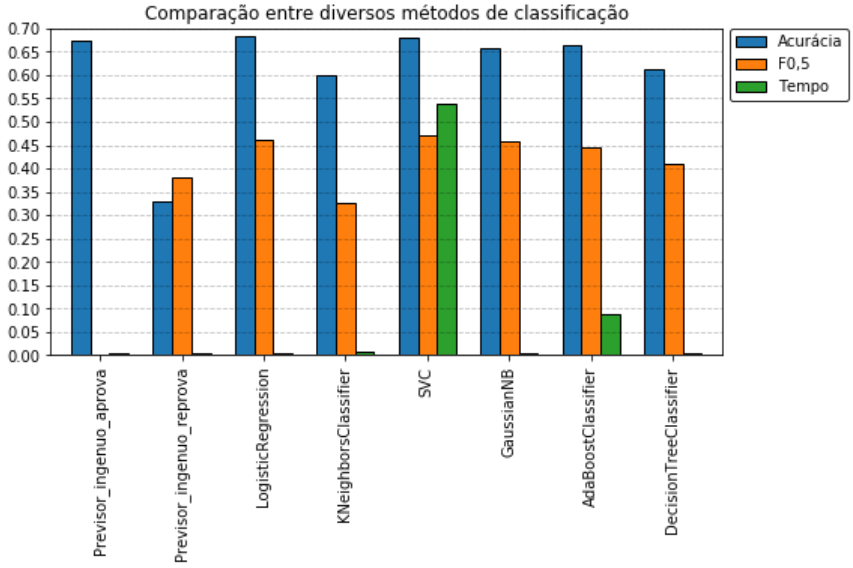
In [ ]:
```
# Cálculo e exibição das métricas para o modelo Regressão Logística
ac = 0
f1 = 0

# Cálculo das métricas
n = 1000
start = time()
for i in range(n):
    logmodel = LogisticRegression()
    res = model_pipeline(X, Yb, logmodel)
    ac += res[0]
    f1 += res[1]
dt = time() - start

# Média das métricas, grava no Data Frame metrics
metrics.loc[logmodel.__class__.__name__] = [round(ac/n, 3), round(f1/n, 3), round(dt/n, 3)]

# Imprime as médias
print(formating.bold + 'Modelo: ' + logmodel.__class__.__name__ + formating.resetbold)
print('Acurácia média = {:.3f}'.format(ac/n))
print('F0,5 scr médio = {:.3f}'.format(f1/n))

# Exibe um exemplo
print(formating.bold + '\nExemplo:' + formating.resetbold)
logmodel = LogisticRegression()
f1 = model_pipeline(X, Yb, logmodel, True)
```

```
Modelo: LogisticRegression
Acurácia média = 0.684
F0,5 scr médio = 0.460

Exemplo:
Matriz de Confusão
 [45  7]
 [21  6]

Acurácia  = 0.646
Precisão  = 0.462
Revocação = 0.222
F0,5 scr  = 0.380
```

Etapa 3.2.d: Análise de acurácia e F0,5 para diferentes modelos de classificação

A seguir, os resultados em forma de tabela e de gráfico dos testes realizados:

	Acurácia	F0,5	Tempo
Previsor_ingenuo_aprova	0.673	0.000	0.003
Previsor_ingenuo_reprova	0.330	0.380	0.003
LogisticRegression	0.684	0.460	0.005
KNeighborsClassifier	0.599	0.325	0.007
SVC	0.680	0.471	0.540
GaussianNB	0.659	0.459	0.004
AdaBoostClassifier	0.664	0.446	0.087
DecisionTreeClassifier	0.614	0.409	0.005



A partir deles, chegou-se às seguintes conclusões:

- Em termos de acurácia, apenas a Regressão Logística e a Máquina de Suporte de Vetor conseguiram superar um modelo que simplesmente prevê que nenhum aluno precisa de reforço. Os modelos Naïve Bayes e AdaBoost obtiveram resultados quase tão bons quanto os dois primeiros. Na verdade, os quatro modelos foram os únicos com acurácia acima de 0,65, contra uma acurácia de 0,67 do previsor ingênuo de aprovação.
- Os quatro modelos citados anateriormente também tiveram os maiores F0,5 Scores, obtendo pelo menos 0,445, contra um F0,5-Score de 0,38 do previsor ingênuo de reprovação. Deve-se lembrar que dar reforço a todos os alunos evita que alunos que seriam reprovados fiquem sem reforço, mas desperdiça muitos recursos com alunos que seriam aprovados sem reforço. Talvez por este motivo, praticamente qualquer modelo (exceto kNN) consegue superá-lo facilmente. **Entretanto, ainda é possível tornar o desempenho dos modelos ainda mais superior ao do previsor ingênuo.**
- O tempo de execução da Máquina de Suporte de Vetor (SVC) é muito maior do que os demais, mas seu desempenho é bastante similar ao da Regressão Logística. Dessa forma, tal modelo não será usado.
- **Conclusão:** Entre os modelos testados, considerando desempenho e tempo, os mais promissores são: Regressão Logística, Naïve Bayes e AdaBoost.

3.3. Refinamento

Nesta etapa, serão estudadas as importâncias de cada atributo para a previsão da necessidade de reforço. Em seguida, os modelos serão testados novamente apenas com atributos mais importantes e as novas métricas serão comparadas com as antigas. Com uma redução no número de atributos, é esperado que os modelos apresentem melhor desempenho em sua previsões.

Etapa 3.3.a: Seleção de atributos

Estudo dos atributos mais importantes. Passos:

- Treinar 250 modelos de cada técnica de classificação;
  - **Regressão Logística:** técnica com um dos melhores resultados obtidos na etapa anterior. Serão usados seus coeficientes para tentar determinar os atributos mais importantes;
  - **AdaBoost:** técnica com bons resultados obtidos na etapa anterior. Serão usadas suas importâncias dos atributos para tentar determinar os atributos mais importantes;
  - **Árvores de Decisão:** tal técnica obteve resultados relativamente fracos na etapa anterior. Entretanto, serão usadas suas importâncias dos atributos para tentar conseguir insights sobre os atributos mas importantes;
  - **Naive Bayes:** apesar de apresentar bons resultados na etapa anterior, o modelo não possui coeficientes ou importâncias dos atributos que ajudem a determinar os atributos mais importantes;
- Tirar a média dos coeficientes ou das importâncias dos atributos de cada técnica;
- Exibir tabela com os maiores coeficientes ou as maiores importâncias dos atributos de cada técnica;
- Exibir gráfico dos coeficientes ou das importâncias dos atributos de cada técnica;
- Escolher os atributos mais importantes.

A seguir, um exemplo da programação para a `Regressão Logística` :

In [ ]:

```
# Seleção de Atributos - Regressão Logística
logcoefs = np.zeros(X.columns.shape[0])

# Cálculo dos coeficientes
n = 250
for i in range(n):
    logmodel = LogisticRegression()
    X_train, X_test, y_train, y_test = train_test_split(X, Yb, test_size=0.2)
    logmodel.fit(X_train, y_train)
    logcoefs += np.array(logmodel.coef_[0])

# Média dos coeficientes, grava em um Data Frame
logcoefs /= n
logmodel_coefs = pd.DataFrame(data={'coef': logcoefs, 'abscoef':abs(logcoefs)},
                              index=X.columns).sort_values(by='abscoef', ascending=False)

# Imprime os maiores coeficientes
print(formating.bold + 'Modelo: ' + logmodel.__class__.__name__ + formating.resetbold)
print(logmodel_coefs[logmodel_coefs.abscoef > 0.4].coef, '\n')

# Gráfico de todos os coeficientes
logmodel_coefs.drop(['coef'], 1).plot.bar(figsize=(10,4), legend=False,
                                          width=0.975, edgecolor='k')
plt.grid(b=True, axis='y', color='k', linestyle='--', alpha=0.25)
plt.title('Coeficientes dos atributos para modelo ' + logmodel.__class__.__name__)
```

A seguir, os resultados em forma de tabela e de gráfico dos testes realizados:

Modelo: LogisticRegression

failures	1.911183
goout	1.436281
age	0.946842
schoolsup	0.722502
higher	-0.714096
studytime	-0.533034
famsup	0.511230
famrel	-0.495615
Mjob_teacher	0.484121
sex	-0.470027
Mjob_health	-0.445088
Fjob_teacher	-0.414713
Walc	-0.406421

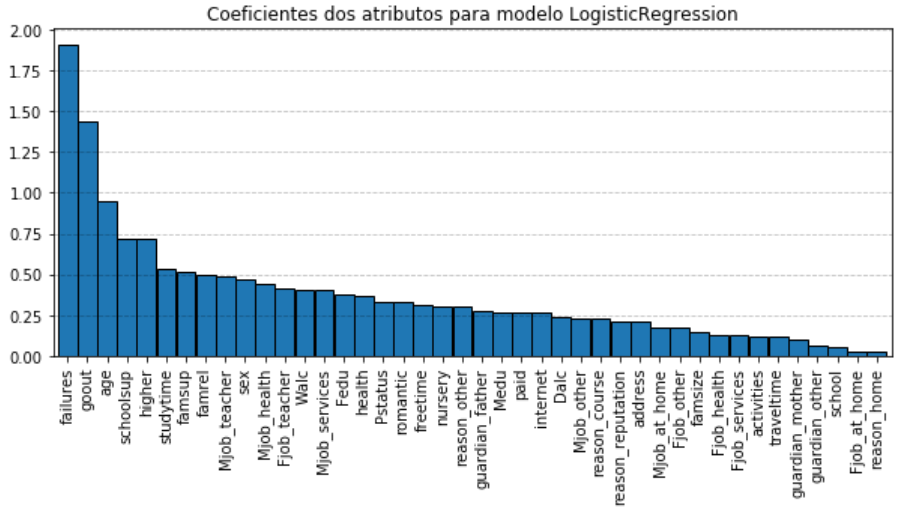
Name: coef, dtype: float64

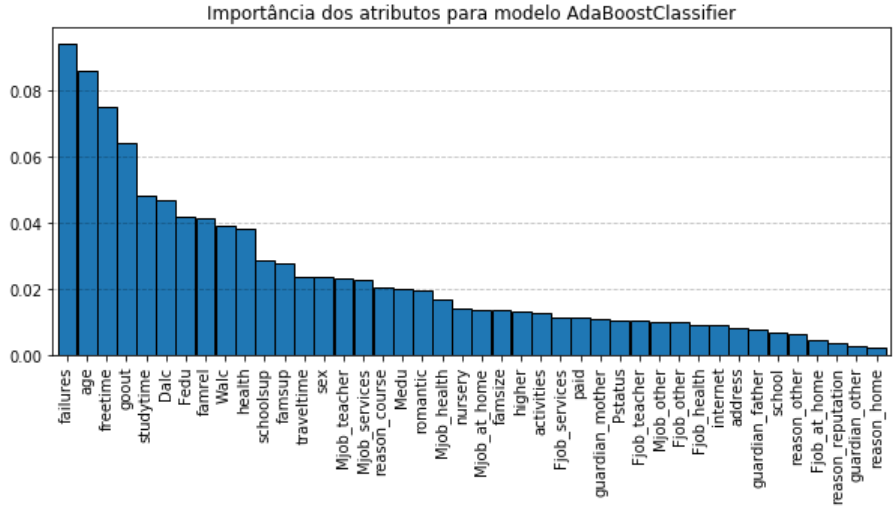
Modelo: AdaBoostClassifier

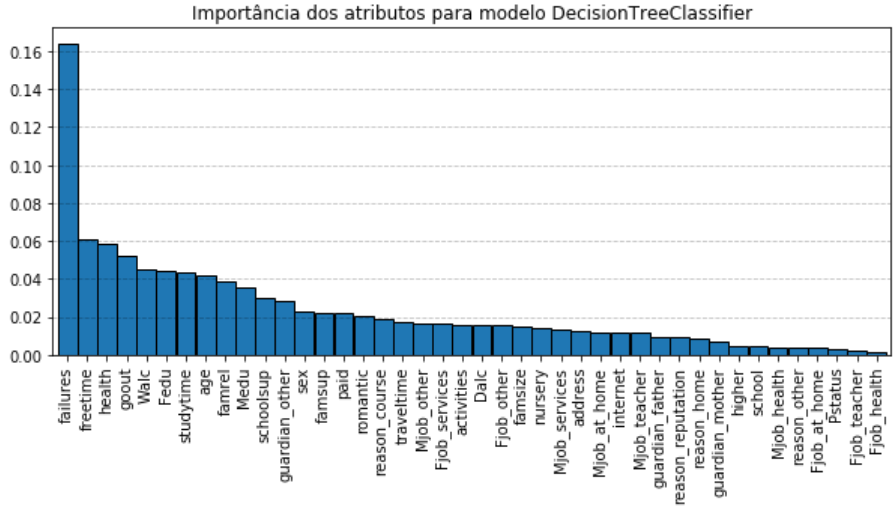
	feat_imp
failures	0.09416
age	0.08600
freetime	0.07512
goout	0.06384
studytime	0.04832
Dalc	0.04672
Fedu	0.04160
famrel	0.04128
Walc	0.03912
health	0.03792

Modelo: DecisionTreeClassifier

	feat_imp
failures	0.164189
freetime	0.060740
health	0.058152
goout	0.052565
Walc	0.045231
Fedu	0.044525
studytime	0.043289
age	0.042025
famrel	0.038444
Medu	0.035179







Análise dos resultados:

- O atributo **schoolsup** denota se o aluno já recebe algum suporte extra da escola e impacta positivamente sobre a necessidade de reforço. Em outras palavras, se o aluno recebe suporte extra, então é mais provável que ele acabe sendo reprovado. Isto pode significar que apenas alunos com dificuldades (e com maiores chances de reprovação) recebem tal suporte. Entretanto, as condições nas quais os alunos são selecionados para tal suporte não estão claras e podem ser subjetivas. Por este motivo, este atributo será **excluído** dos modelos de previsão;
- O atributo **sex** tem o 10º maior coeficiente da Regressão Logística, mas tem importância bem menor nos demais modelos. Por este motivo, ele também será **excluído** dos modelos de previsão;
- Os 10 atributos com maiores coeficientes da Regressão Logística, exceto os dois anteriores (schoolsup e sex) serão mantidos: failures, goout, age, higher, studytime, famsup, Mjob_teacher, famrel;
- Os atributos **freetime** e **health** estão entre os mais importantes dos modelos AdaBoost e DecisionTree e, portanto, serão mantidos;
- O atributo **Dalc** é o quinto mais importante do AbaBoost e também será mantido;
- Todos os demais atributos serão excluídos.

Os seguintes atributos foram mantidos:

```
01) failures            02) goout               03) age                 04) higher
05) studytime           06) famsup              07) Mjob_teacher        08) famrel
09) freetime            10) health              11) Dalc
```

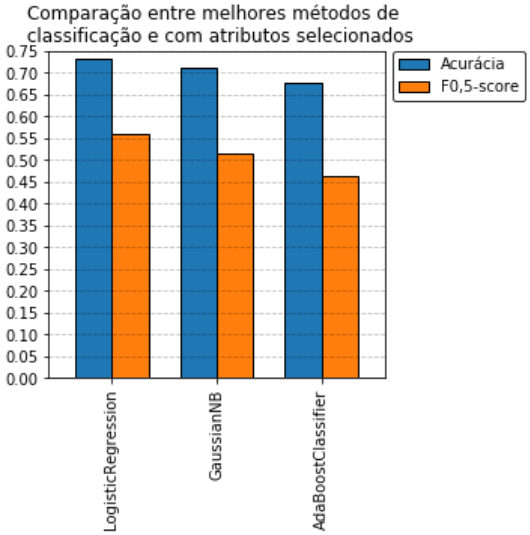
Etapa 3.3.b: Comparação acurácia e F0,5 para atributos selecionados

A Etapa `3.2.c` será repetida para as técnicas de classificação com melhores resultados, e apenas com os atributos selecionados na Etapa `3.3.a`. Passos:

- Criar variável **X2** com os atributos selecionados;
- Criar um Data Frame chamado **metrics2** para armazenar as métricas (acurácia e F0,5-score) de cada técnica;
- Aplicar diversos algortimos e gravar seus resultados em **metrics2**:
  - Calcula a média da acurácia e do F0,5-score para 1000 previsões de cada técnica;
  - Grava e imprime as médias;
  - Calcula um modelo usando **verbose** para imprimir matriz de confusão, precisão e revocação, como exemplo.
- Exibir os resultados em uma tabela e em um gráfico;
- Analisar os resultados e tirar conclusões.

A programação desta etapa pode ser encontrada no arquivo `code`. A seguir, estão os resultados em forma de tabela e de gráfico:

	Acurácia	F0,5-score
LogisticRegression	0.733	0.560
GaussianNB	0.712	0.515
AdaBoostClassifier	0.677	0.464



Análise dos resultados:

- Ao se selecionar apenas os atributos mais importantes ('failures', 'goout', 'age', 'higher', 'studytime', 'famsup', 'Mjob_teacher', 'famrel', 'freetime', 'health' e 'Dalc'), os três melhores modelos de classificação determinados na etapa 4 (LogisticRegression, GaussianNB e AdaBoostClassifier) melhoraram significativamente seu desempenho.
- A **Regressão Logística** continua sendo a melhor técnica de modelagem, tendo sua acurácia elevada de 0,68 para 0,73 e seu F0,5-score aumentado de 0,46 para aproximadamente 0,56. Seu desempenho nestas condições é o melhor obtido neste trabalho.

---

4. Resultados

4.1. Modelo de avaliação e validação

O modelo final para resolver o presente problema tem as seguintes caracerísticas:

- Gerado através do algoritmo de **Regressão Logística**, com os parâmetros default do *scikitlearn*;
- Utiliza como atributos as variáveis: *'failures', 'goout', 'age', 'higher', 'studytime', 'famsup', 'Mjob_teacher', 'famrel', 'freetime', 'health' e 'Dalc'*;
- Utiliza como alvo o desempenho final do estudantes (**0** se nota G3 >= 10 / **1** se nota G3 < 10);
- Trenado com um subconjunto de treino com 80% das instâncias do conjunto original.

Este modelo foi escolhido por conta de seu desempenho superior nos testes realizados:

- Apresenta **acurácia média** (em 1000 amostras) de 0,733 - valor 9% acima do previsor ingênuo que aprova todos os alunos (0,673), 4% acima do benchmark (0,706 obtida pelo algoritmo de Máquina de Vetor de Suporte na configuração C do `item 2.4`), e 3% acima do segundo melhor modelo testado (0,712 obtida pelo Naive Bayes Gaussiano sob as mesmas condições).
- Apresenta $F_{\beta=0,5} - score$ **médio** (em 1000 amostras) de 0,560 - valor 48% acima do previsor ingênuo que reprova todos os alunos (0,380), e 9% acima do segundo melhor modelo testado (0,515 obtido pelo Naive Bayes Gaussiano sob as mesmas condições).
- Apresenta **tempo médio** para treino e previsão (em 1000 amostras) de 0,005 s - aproximadamente igual ao tempo médio obtido pelo Naive Bayes Gaussiano, k Vizinhos Mais Próximos e Árvores de Decisão e significativamente menor do que os tempos obtidos pelo AdaBoost (0,087 s) e pela Máquina de Vetor de Suporte (0,540 s).

É importante ressaltar que as métricas foram usadas para avaliar previsões de 1000 modelos diferentes acerca dos subconjuntos de testes. Em outras palavras, as métricas se referem ao desempenho dos modelos na previsão de dados que eles não conheciam durante o treinamento. Além disso, a média de 1000 amostras garante robustez aos resultados.

4.2. Justificativa

A referência deste projeto indica apenas valores de acurácia dos modelos testados. Mesmo assim, ao diminuir a quantidade de atributos de entrada, foi possível obter uma acurácia de 0,733 em 1000 amostras - 4% acima do benchmark (observe que há um Intervalo de Confiança de 0,706 ± 0,004).

Como as métricas obtidas pelo modelo final são bem superiores às obtidas por previsores ingênuos, tem-se confiança em afirmar que o problema encontrou uma solução no mínimo satisfatória.

---

5. Conclusão

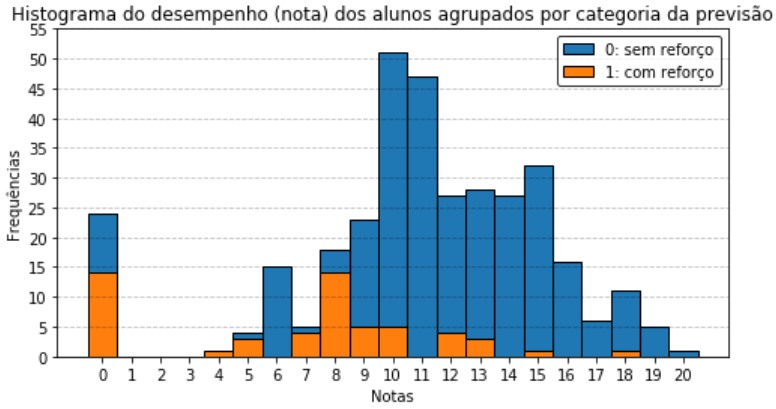
5.1. Forma livre de visualização

O modelo final foi usado para gerar previsões acerca de todos os dados do dataset. O código para treino do modelo e eração das visualizações está no arquivo `code` .

A partir de tais previsões, gerou-se:

1. Um resumo com as principais métricas (matriz de confusão, acurácia, precisão, revocação e F0,5-scr);
2. Um histograma de notas distinguindo alunos classificados como sem reforço/com reforço.

```
Modelo Final: LogisticRegression
Métricas para o dataset completo:

Matriz de Confusão
 [251  14]
 [89 41]

Acurácia  = 0.739
Precisão  = 0.745
Revocação = 0.315
F0,5 scr  = 0.586
```



Pode-se observar que boa parte dos alunos com nota abaixo de 10 foram identificados corretamente pelo modelo, enquanto que uma quantidade muito baixa de alunos com notas a partir de 10 foram classificados incorretamente.

Além disso, é notável como o modelo evita classificar os alunos como elegíveis para algum programa de reforço para tentar manter a precisão elevada e, consequentemente, o $F_{\beta=0,5} - score$. Como alta precisão significa baixa quantidade Falsos Positivos, pode-se concluir que o modelo mais eficiente em termos de custos é aquele que prevê poucas necessidades de reforço.

5.2. Reflexão

Este processo envolveu as seguintes etapas:

1. **Análise das variáveis e pré-processamento:** Esta foi uma etapa relativamente simples, uma vez que não haviam dados nulos em nenhuma instância do conjunto de dados e uma vez que as variáveis não eram muito complexas para se trabalhar. No máximo, foi necessário binarizar, alterar a escala e usar codificação one-hot. Em datasets mais difíceis, pode-se ter a necessidade de preencher nulos ou de ter que lidar com variáveis como localização geográfica.
2. **Escolha das métricas:** a referência usada neste projeto empregou apenas a acurácia como métrica. Como se sabe, em datasets desbalanceados como este (com 2/3 dos alunos aprovados), tal métrica não apresenta bons resultados. Não é por acaso que em certas configurações de teste, nenhum modelo conseguiu bater um previsor ingênuo em ermos de acurácia. Dito isso, um grande desafio foi estabelecer a métrica mais adequada sem ter uma "seguna opinião" de um benchmark. Inicialmente, cogitei um F-score com β = 1. Entretanto, julguei que um bom modelo deveria presar mais pela precisão e evitar desperdiçar recursos financeiros e humanos dando reforço a alunos que não precisam. Evidentemente, se o governo dispusesse de recursos ilimitados, poderia usá-los para simplesmente aumentar o nível da educação como um todo. Por este motivo, julguei que um β = 0,5 é mais condizente com uma aplicação prática e realista.
3. **Escolha dos modelos:** aqui, eu decidi não inovar muito e busquei modelos bem conhecidos e consolidados no meio de Machine Learning, e por isso preferi os algoritmos básicos e populares do scikitlearn. O trabalho de benchmark usou redes neurais, mas julguei que este trabalho se tornaria muito complexo se eu fosse aplicar conhecimentos mais recentes de Deep Learning. Por fim, avaliei que esforços como GridSearch trariam ganhos marginais em termos de desempenho ao custo de maior esforço e tempo, e por isso exclui tal etapa.
4. **Seleção de atributos:** foi a etapa mais gratificante. Após gastar muito esforço nas etapas anteriores sem conseguir resultados muito expressivos, quando diminui a quantidade de atributos, consegui melhorar dasticamente o desempenho dos modelos. Não me limitei aqui a aplicar a `regra do cotovelo` e selecionar apenas os atributos mais significativos antes da curva se estabilizar. Ao invés disto, busquei uma lista maior de atributos que estavam sendo indicados como importantes por algoritmos diferentes. Eu cheguei a testar listas finais com mais e com menos atributos, mas esta foi a que trouxe melhores resultados. Creio ter chegado próximo ao limite do que é possível fazer com a seleção de atributos.
5. **Robustez dos resultados:** eu notei que a escolha dos conjuntos de treino e de teste exercia forte influência sobe as métricas, e por isto decidi fazer médias entre 1000 treinos, cada um com um conjunto diferente de treino. Como consequência, a repetibilidade dos resultados aumentou e eu pude ter mais confiança de que as diferenças entre as métricas obtidas nos resultados eram significativas e não frutos do acaso.

Como resultado, obtive um modelo que consegue encontrar boa parte dos alunos que falharão no fim do ano e preservar boa parte dos que serão em sucedidos. Melhorias ainda podem ser feitas, mas este modelo oferece um meio confiável de prever necessidade de reforço, garantindo certa acionalidade no uso de recursos.

5.3. Melhorias

Posso indicar melhorias em todas as etapas da seção anterior:

1. **Análise das variáveis e pré-processamento:** Pode-se investigar se existem formas melhores de alterar a escala das variáveis numéricas e de se codificar as variáveis categóricas nominais. Por exemplo, pode-se normalizar as variáveis numéricas ((x - média) / desvio padrão), ou buscar alternativas ao one-hot. Também pode-se trabalhar com um alvo multi-classes, tentando prever, por exemplo, se um aluno ficará com nota zero. Isto seria relevante na medida em que um aluno com zero é, com certeza, um desistente, e por isso não precisará de reforço. Novas formas de se determinar o alvo certamente trariam melhores modelos.
2. **Escolha das métricas:** Pode-se comparar diferentes valores de β para tentar obter algum novo conhecimento.
3. **Escolha dos modelos:** Pode-se aplicar algoritmos mais complexos como Redes Neurais Profundas e/ou aplicar GridSearch para encontrar parâmetros ótimos dos modelos.
4. **Seleção de atributos:** Pode-se buscar outros modelos que possuam o método `feature_importance` para tentar melhorar ainda mais a escolha de atributos.
5. **Robustez dos resultados:** Pode-se calcular o desvio padrão as 1000 amostras (além de calcular apenas a média) e determinar os Inervalos de Confiança das métrias empregadas.

---

6. Referências Bibliográfias

01 Cortez, P. e Silva, A. **Using Data Mining to Predict Secondary School Student Performance.** In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7. [WEB link] (http://www3.dsi.uminho.pt/pcortez/student.pdf)

02 Kotsiantis, S.; Pierrakeas, C. e Pintelas, P. **Predicting Students' Performance in Distance Learning Using Machine Learning Techniques.** Applied Artificial Intelligence (AAI), 18, no. 5, 2004, 411–426. [WEB link] (https://pdfs.semanticscholar.org/ca32/7f56f4290809d7c243b57c97bb2eb0916ff1.pdf)

03 UC Irvine Machine Learning Repository. **Student Performance Data Set.** [WEB link] (https://archive.ics.uci.edu/ml/datasets/Student+Performance)

04 GeeksforGeeks. **Confusion Matrix in Machine Learning.** [WEB link] (https://www.geeksforgeeks.org/confusion-matrix-machine-learning/)

05 Mueller, A.C.; Guido, S. **Introduction to Machine Learning with Python.** O'Reilly, 2016.

06 Schapire, R. **A Boosting Tutorial.** [WEB link] (https://www.csie.ntu.edu.tw/~mhyang/course/u0030/papers/schapire.pdf)

07 GeeksforGeeks. **Data Preprocessing for Machine learning in Python.** [WEB link] (https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/)

08 Kaggle. **Using Categorical Data with One Hot Encoding.** [WEB link] (https://www.kaggle.com/dansbecker/using-categorical-data-with-one-hot-encoding)