PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Lucas Rocha Rodrigues

ESTIMATIVA DO DESEMPENHO ACADÊMICO UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA

Belo Horizonte
2021

Lucas Rocha Rodrigues

ESTIMATIVA DO DESEMPENHO ACADÊMICO UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte
2021

SUMÁRIO

1. Introdução

1.1. Contextualização

Durante os anos de 2005-2006, os autores Cortez e Silva (2008), com muito mérito, realizaram uma ampla pesquisa em duas escolas secundárias de Portugal (equivalente a uma escola de ensino médio no Brasil) com o objetivo de descobrir e entender fatores que afetam o desempenho acadêmico dos alunos utilizando técnicas de *Business Inteligence*. Resumidamente, a pesquisa consistiu em reunir, através da aplicação de questionário, uma série de dados demográficos, sociais, comportamentais, emocionais, e estudantis, e uni-los a dados escolares desses alunos, tais como as notas obtidas em avaliações de português e matemática, faltas no ano e quantidade de reprovações anteriores na disciplina.

Os dados coletados pelos autores no estudo descrito acima foram publicamente disponibilizados e representam uma oportunidade de se buscar novos insights a respeito dos fatores que afetam o desempenho acadêmico, bem como de validar o que já foi concluído. Portanto, espera-se com o trabalho atual expandir e enriquecer o entendimento da aplicação de técnicas de Ciência de Dados na previsão de desempenho acadêmico. Isso será feita em três frentes: (1) execução de uma vasta análise exploratória dos dados; (2) aplicação de técnicas de engenharia de atributos para obtenção de novos e relevantes *features*; e (3) utilização do classificador Árvore de Decisão para efetuar a previsão de desempenho, procurando otimizar alguns de seus hiperparâmetros e exibir as acurácias obtidas.

1.2. O problema proposto

Busca-se entender e prever o desempenho acadêmico de alunos da educação secundária de Portugal com base em uma série de atributos demográficos, sociais, comportamentais e estudantis utilizando técnicas da Ciência de Dados. Abaixo se encontram as respostas às 5 perguntas presentes na técnica dos 5WS para melhor se entender e dimensionar o problema em questão.

***Why*? /Por que?** A educação é um dos pilares da sociedade e uma fonte de significativos investimentos por parte de governos e cidadãos, portanto, esforços para melhor compreendê-la e otimizá-la são sempre bem-vindos, e a Ciência de Dados oferece técnicas com potencial de contribuir significativamente com esses esforços.

***Who*? /Quem?** Serão analisados dados de alunos de duas escolas secundárias públicas (equivalente ao ensino médio brasileiro) da região de Alentejo em Portugal.

***What*? /O que?** O objetivo da análise é investigar a influência de diversos fatores no desempenho acadêmico dos alunos e avaliar a acurácia de modelos de aprendizado de máquina na previsão do desempenho.

***Where*? /Onde?** Embora as análises tenham sido conduzidas para alunos de escolas Portuguesas, as técnicas empregadas bem como os insights gerados podem ser aplicados à qualquer região, respeitando-se aspectos e características locais.

***When*? /Quando?** A análise foi feita com base nos dados foram levantados no ano letivo de 2005-2006.

2. Coleta de Dados

Os dados originalmente coletados pelos autores Cortez e Silva (2008) foram publicamente disponibilizados na plataforma *Kaggle* (link nas referências). Tratam-se de dados estruturados e tabulares no formato CSV (*Comma Separated Values)*. Foram disponibilizados dois arquivos, um para a matéria matemática e outro para português. Cada linha contém os atributos de um aluno, e em muitos casos o mesmo aluno possui registros nos dois *datasets*, ou seja, há informações sobre seu aluno letivo de matemática e português. Para ambos arquivos os atributos presentes são os seguintes.

Tabela 1: Descrição dos atributos

Atributo	Descrição	Tipo
school	Qual das duas escolas o aluno frequenta	String/Object

sex	Sexo do aluno	String/Object
age	Idade do aluno (entre 15 e 22 anos)	Integer
address	Se o aluno mora na cidade ou zona rural	String/Object
famsize	Se a família do aluno possui mais ou menos que três pessoas	String/Object
Pstatus	Pais vivem juntos ou separados	String/Object
Medu	Escolaridade da mãe (entre 0 e 4, sendo que 1: educação primária, 2: quinta a nona série, 3: educação secundária, 4: educação superior)	Integer
Fedu	Escolaridade do pai (entre 0 e 4, sendo que 1: educação primária, 2: quinta a nona série, 3: educação secundária, 4: educação superior)	Integer
Mjob	Tipo de trabalho da mãe (professora, área da saúde, serviços civis, dona de casa, outro)	String/object
Fjob	Tipo de trabalho do pai (opções: professor, área da saúde, serviços civis, dono de casa, outro)	String/object
reason	Razão pela qual escolheu a escola (tem o curso que o aluno deseja, perto de casa, reputação, outro motivo)	String/object
guardian	Responsável pelo aluno	String/object
traveltime	Tempo gasto para chegar à escola (1: < 15 min, 2: 15-30 min, 3: 30 min - 1 hora, 4: > 1 hora)	Integer
studytime	Tempo semanal de estudo da disciplina (1: < 2 horas, 2: 2-5 horas, 3: 5-10 horas, 4: > 10 horas)	Integer
failures	Número de reprovações anteriores na disciplina	Integer
schoolsup	Se o aluno recebe suporte educacional extra da escola	String/object
famsup	Se a família recebe suporte educacional	String/object
paid	Se o aluno pagou aulas particulares da matéria	String/object
activities	Se o aluno faz atividades extracurriculares	String/object
nursery	Se o aluno cursou o maternal	String/object
higher	Se o aluno pretende cursar faculdade	String/object
internet	Se o aluno possui acesso a internet em casa	String/object
romantic	Se o aluno encontra-se em um relacionamento amoroso	String/object
famrel	Nota (de 1 a 5) para relação familiar do aluno	Integer

freetime	Nota (de 1 a 5) para a disponibilidade de tempo livre após a escola	Integer
goout	Nota (de 1 a 5) para a frequência com que o aluno sái com os amigos	Integer
Dalc	Nota (de 1 a 5) para o consumo de álcool durante os dias de semana	Integer
Walc	Nota (de 1 a 5) para o consumo de álcool durante os dias de fim de semana	Integer
health	Nota (de 1 a 5) em que o aluno avalia sua saúde	Integer
absences	Número de faltas que o aluno teve naquela disciplina	Integer
G1	Nota na prova do primeiro período (de 0 a 20)	Integer
G2	Nota na prova do segundo período (de 0 a 20)	Integer
G3	Nota final na dispciplina (determinada por uma terceira prova) (de 0 a 20)	Integer

3. Processamento/Tratamento de Dados

3.1. Considerações iniciais

Antes de iniciar a explicação dos passos realizados, vale informar que todo processamento aqui mostrado, bem como todas próximas etapas da análise feita nesse trabalho, foram feitas utilizando a linguagem Python em *Jupyter Notebooks* lançados a partir do software *Anaconda*, que já disponibiliza uma série de bibliotecas Python para serem facilmente importadas e utilizadas.

Vale dizer também que toda manipulação dos dados será feita utilizando a biblioteca *Pandas*, pela facilidade que oferece para se lidar com dados estudo. Outras bibliotecas utilizadas no trabalho para tarefas mais específicas serão apresentadas em suas respectivas seções.

Aqui cabe também apresentar os dois principais *datasets* utilizados no trabalho, que contém todas informações dos alunos nas matérias português e matemática. A figura a seguir demonstra a criação dos *DataFrames* de cada matéria a partir de seus arquivos .csv.

Figura 1: Criação dos DataFrames para cada matérias

```
math = pd.read_csv('student-mat.csv')
port = pd.read_csv('student-por.csv')
```

3.2. Checagem dos *datasets* quanto a valores nulos

Para se conferir a presença ou não de valores nulos em cada registro dos *datasets* utilizou-se o método dos *DataFrames .info()*. Obteve-se a seguintes saídas abaixo.

Figura 2: Saídas do método .info() em cada dataset

```
math.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
```

```
port.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 649 entries, 0 to 648
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      649 non-null    object
 1   sex         649 non-null    object
 2   age         649 non-null    int64
 3   address     649 non-null    object
 4   famsize     649 non-null    object
 5   Pstatus     649 non-null    object
 6   Medu        649 non-null    int64
 7   Fedu        649 non-null    int64
 8   Mjob        649 non-null    object
 9   Fjob        649 non-null    object
 10  reason      649 non-null    object
 11  guardian    649 non-null    object
 12  traveltime  649 non-null    int64
 13  studytime   649 non-null    int64
 14  failures    649 non-null    int64
 15  schoolsup   649 non-null    object
 16  famsup      649 non-null    object
 17  paid        649 non-null    object
 18  activities  649 non-null    object
 19  nursery     649 non-null    object
 20  higher      649 non-null    object
 21  internet    649 non-null    object
 22  romantic    649 non-null    object
 23  famrel      649 non-null    int64
 24  freetime    649 non-null    int64
 25  goout       649 non-null    int64
 26  Dalc        649 non-null    int64
 27  Walc        649 non-null    int64
 28  health      649 non-null    int64
 29  absences    649 non-null    int64
 30  G1          649 non-null    int64
 31  G2          649 non-null    int64
 32  G3          649 non-null    int64
dtypes: int64(16), object(17)
memory usage: 167.4+ KB
```

Como se pode ver, nenhuma das colunas apresenta valores nulos. Acredita-se que isso se dá, pois como esses dados serviram de base para o estudo sociológico dos autores Cortez e Silva (2008), apenas os registros com todas informações foram considerados.

3.3. Codificação dos atributos categóricos com *Label Encoding*

Como se pode ver na figura acima, vários dos atributos possuem entradas categóricas, contudo, algumas das técnicas a serem empregadas na sequência desse trabalho – tais como a obtenção dos valores de informação mútua dos atributos e a clusterização - requerem que os valores de entrada sejam numéricos. Para se resolver isso, será feita a codificação desses atributos utilizando a técnica de *Label Encoding*, que basicamente consiste em transformar cada categoria em um valor numérico inteiro. Todo o procedimento de codificação com *Label Encoding* pode ser visto no *Jupyter Notebook* 01_Informação mútua e Correlação presente repositório mencionado no fim desse trabalho.

Para se fazer a codificação, primeiramente se obtém uma lista com todas as variáveis categóricas do *dataset*, o que pode ser feito com o método DataFrame.select_dtypes('object'). A seguir, mostra-se a aplicação desse método, bem como o as colunas categóricas do dataset antes da transformação.

Figura 3: Colunas categóricas do dataset

```
# Obtendo lista das variáveis categóricas
catFeatures = math.select_dtypes('object').columns.to_list()
print(catFeatures)
```

```
['school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activi
ties', 'nursery', 'higher', 'internet', 'romantic']
```

```
math[catFeatures].head()
```

	school	sex	address	famsize	Pstatus	Mjob	Fjob	reason	guardian	schoolsup	famsup	paid	activities	nursery	higher	internet	romantic
0	GP	F	U	GT3	A	at_home	teacher	course	mother	yes	no	no	no	yes	yes	no	no
1	GP	F	U	GT3	T	at_home	other	course	father	no	yes	no	no	no	yes	yes	no
2	GP	F	U	LE3	T	at_home	other	other	mother	yes	no	yes	no	yes	yes	yes	no
3	GP	F	U	GT3	T	health	services	home	mother	no	yes	yes	yes	yes	yes	yes	yes
4	GP	F	U	GT3	T	other	other	home	father	no	yes	yes	no	yes	yes	no	no

Na sequência, utiliza-se o método *fit_transform()* do objeto *LabelEncoder* do módulo de pre-processamento da biblioteca *Scikit Learn* para efetuar a transformação. A figura a seguir mostra a aplicação do método em cada uma das colunas categóricas e o *dataset* resultante.

Figura 4: Transformação dos dados categóricos com *LabelEncoder*

```
from sklearn.preprocessing import LabelEncoder
# Criando cópia dos dados
mathModificada = math.copy()
# Criando o LabelEncoder
label_encoder = LabelEncoder()
# Alterando as colunas categóricas
for col in catFeatures:
    mathModificada[col] = label_encoder.fit_transform(math[col])
# Conferindo o resultado
mathModificada[catFeatures].head()
```

	school	sex	address	famsize	Pstatus	Mjob	Fjob	reason	guardian	schoolsup	famsup	paid	activities	nursery	higher	internet	romantic
0	0	0	1	0	0	0	4	0	1	1	0	0	0	1	1	0	0
1	0	0	1	0	1	0	2	0	0	0	1	0	0	0	1	1	0
2	0	0	1	1	1	0	2	2	1	1	0	1	0	1	1	1	0
3	0	0	1	0	1	1	3	1	1	0	1	1	1	1	1	1	1
4	0	0	1	0	1	2	2	1	0	0	1	1	0	1	1	0	0

4. Análise e Exploração dos Dados

4.1. Considerações iniciais

A seguir será mostrada a exploração feita nos dados. Aqui serão apresentados os principais trechos de código utilizados, mas para visualização do código completo pode-se consultar o *Jupyter Notebook* - Análise Exploratória de Dados, para o qual o link está disponibilizado no final deste trabalho.

4.2. Análise Exploratória

Buscou-se apresentar de forma resumida algumas características gerais do desempenho e comportamento do aluno frente a cada disciplina, tal como a nota média alcançada em cada prova, o tempo médio de estudo reportado, a quantidade média de reprovações anteriores e faltas, o percentual de alunos que recorreram a aulas particulares, e por fim, a taxa de aprovação na disciplina. Para tal, o primeiro passo foi a construção da tabela abaixo (detalhes da construção no *notebook* – Análise Exploratória Dados).

Tabela 2: Informações gerais do desempenho em cada disciplina

	G1	G2	G3	studytime	absences	failures	percent_paid	pass_rate
math	10,909	10,714	10,415	2,035	5,709	0,334	0,458	0,671
port	11,399	11,570	11,906	1,931	3,659	0,222	0,060	0,846

Para se plotar as informações da tabela acima, fez-se uso do método *subplot()* do objeto *pyplot* da biblioteca *matplotlib*. Também utilizou-se método *plot()* do próprio DataFrame. O código utilizado e os resultados obtidos estão mostrados abaixo.

Figura 5: Código utilizado para plotar as informações de desempenho

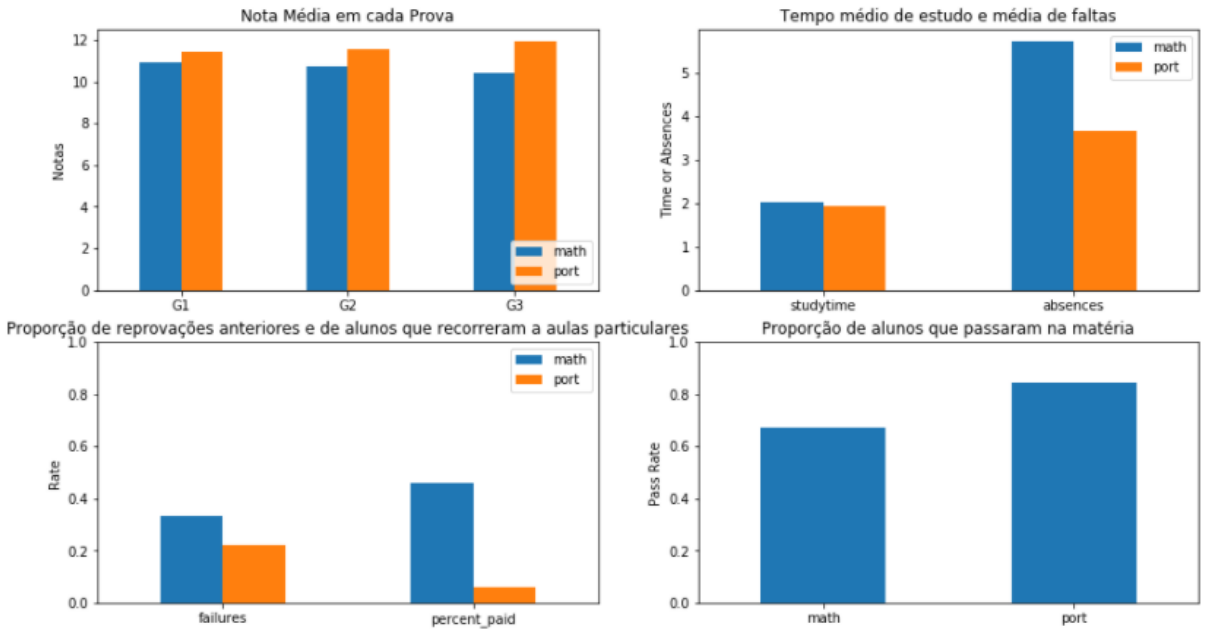
```
# Criando uma figura com 4 gráficos (dispostos numa matriz 2x2)
fig, axs = plt.subplots(2, 2, figsize=(15, 10))
# Especicando o conteúdo de cada gráfico.
descricao[['G1','G2','G3']].transpose().plot(kind = 'bar', ax = axs[0][0])
descricao[['studytime','absences']].transpose().plot(kind = 'bar', ax = axs[0][1])
descricao[['failures','percent_paid']].transpose().plot(kind = 'bar', ax = axs[1][0])
descricao['pass_rate'].transpose().plot(kind = 'bar', ax = axs[1][1])

# Detalhando a construção de cada gráfico
    # Gráfico 1x1
axs[0][0].set_ylabel('Notas')
axs[0][0].legend(loc = 'lower right')
axs[0][0].set_title('Nota Média em cada Prova')
axs[0][0].set_xticklabels(labels = ['G1', 'G2','G3'], rotation = 0)
    # Gráfico 1x2
axs[0][1].set_ylabel('Time or Absences')
axs[0][1].set_title('Tempo médio de estudo e média de faltas')
axs[0][1].set_xticklabels(labels = ['studytime', 'absences'], rotation = 0)
    # Gráfico 2x1
axs[1][0].set_ylabel('Rate')
axs[1][0].set_xticklabels(labels = ['failures','percent_paid'], rotation = 0)
axs[1][0].set_ylim(0,1)
axs[1][0].set_title('Proporção de reprovações anteriores e de alunos que recorreram a aulas particulares')
    # Gráfico 2x2
axs[1][1].set_ylabel('Pass Rate')
axs[1][1].set_xticklabels(labels = ['math','port'], rotation = 0)
axs[1][1].set_ylim(0,1)
axs[1][1].set_title('Proporção de alunos que passaram na matéria')

fig.show
```

Figura 6: Informações de desempenho em cada disciplina - Parte 1

```
<bound method Figure.show of <Figure size 1080x576 with 4 Axes>>
```

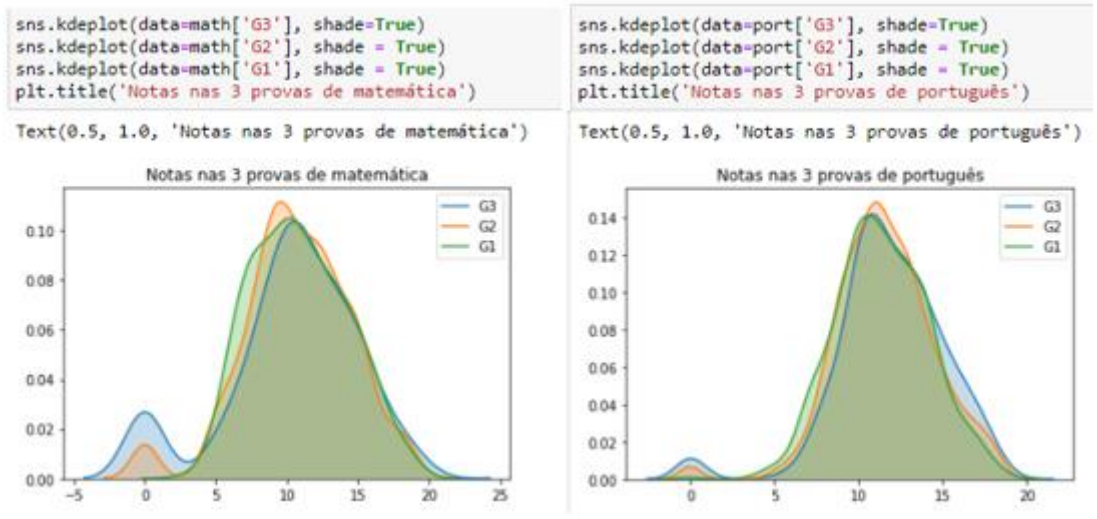


Alguns insights que podem ser tirados das figuras acima:

- O desempenho em português é melhor do que em matemática, possivelmente apontando uma maior dificuldade dos alunos em matemática.
- O ponto acima é corroborado pela proporção muito maior de alunos que recorrem aulas particulares de matemática em relação aos que recorrem a aulas particulares de português.
- Não há diferença significativa entre o tempo de estudo médio para cada matéria.

A seguir, mostra-se a distribuição de densidade de probabilidade das notas dos alunos nas três provas de cada matéria. Para se fazer tal representação se utilizou o gráfico *kdeplot()* da biblioteca *Seaborn*, uma rica biblioteca focada em gráficos para análise exploratória de dados.

Figura 7: Desempenho nas 3 provas de cada matéria



Nota-se que de maneira geral, não há significativa diferença no desempenho global nas três provas, embora, tanto para português quanto para matemática a terceira prova esteja levemente deslocada para a direita, ou seja, os alunos tendem a ir melhor na terceira prova. As curvas das três provas estarem sobrepostas é um ponto de preocupação, pois pode indicar não estar havendo evolução dos alunos com o decorrer do ano letivo.

Segundo a documentação do dataset, 382 alunos possuem registros em ambas as bases. A forma de identificar esses alunos é avaliando se seus atributos sociais e demográficos são iguais nas duas bases. Procurou-se fazer isso com o código abaixo, que utiliza o método *.merge()* dos *DataFrames*, que une duas bases de acordo com os atributos passados ao parâmetro *on*. Isso foi feito para se avaliar

a associação entre a nota que o aluno tira em português e sua nota em matemática. Em seguida se plotou a associação entre as notas finais de cada prova utilizando os comandos *jointplot()* e *lmplot()*, ambos da biblioteca *seaborn.*

Figura 8: Criação de um dataset que reune notas de alunos que cursaram as duas matérias

```
# Renomeando as colunas com notas das provas, para não haver confusão no novo dataset
math2 = math.rename(columns={'G1': 'G1_math',
                             'G2': 'G2_math',
                             'G3': 'G3_math'})
port = port.rename(columns = {'G1': 'G1_port',
                              'G2': 'G2_port',
                              'G3': 'G3_port'})
print(math.shape)
print(port.shape)
```

```
(395, 35)
(649, 33)
```

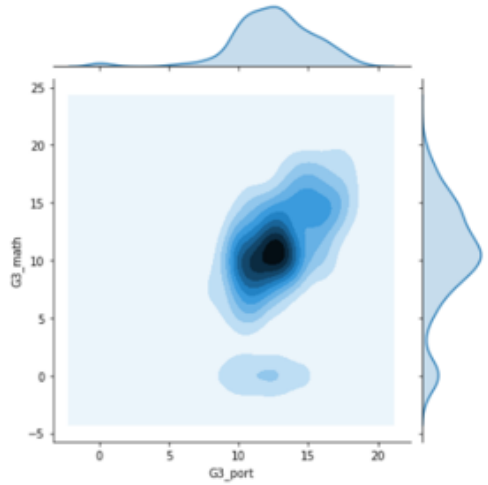
```
#Criando lista com atributos com dados dos alunos (que independem da matéria em questão)
atributos = ["school","sex","age","address","famsize","Pstatus",
             "Medu","Fedu","Mjob","Fjob","reason","nursery","internet"]
```

```
#Fazendo a união baseada em todos atributos
notas = math2.merge(port, on = atributos, how = 'left')
#Simplemente remover ("drop") a linha inteira caso o valor de algum valor seja "NAN"
notas = notas.dropna()
notas = notas.reset_index()
```

Figura 9: Relação entre a nota de português e nota de matemática

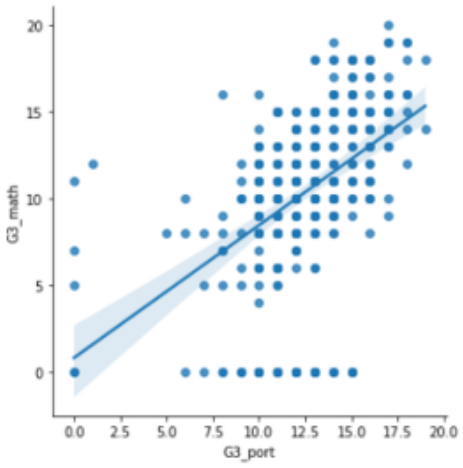
```
sns.jointplot(x = notas['G3_port'], y = notas['G3_math'], kind="kde")
```

```
<seaborn.axisgrid.JointGrid at 0x6eb95531d0>
```




```
sns.lmplot(x="G3_port", y="G3_math", data=notas)
```

```
<seaborn.axisgrid.FacetGrid at 0x6eb967fdd8>
```



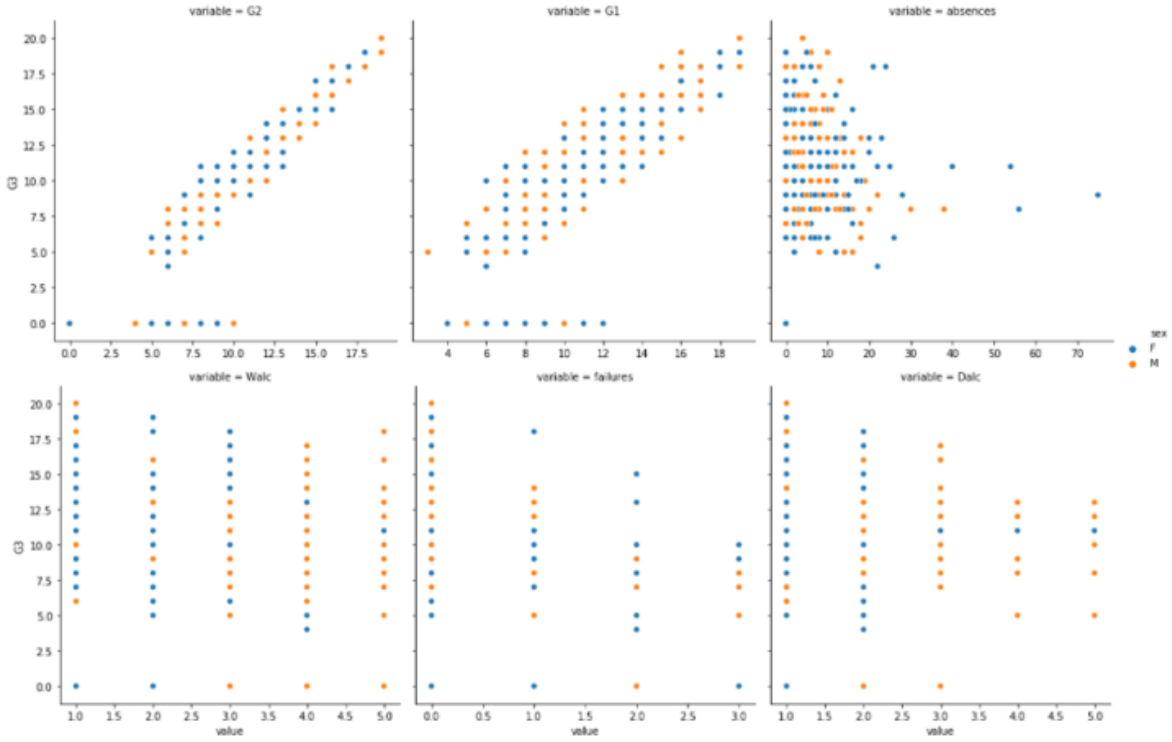
Os gráficos acima demonstram que a nota do aluno em uma matéria está fortemente associada com a nota na outra. Também se percebe a maior tendência do aluno ter uma nota melhor em português, o que é demonstrado pela reta de regressão do *lmplot*() ter um coeficiente angular menor do que 1, também por boa parte da distribuição de probabilidade demonstrada pela *jointplot()* estar à direita da nota 10 em português, mas haver uma parte significativa abaixo da nota 10 em matemática.

A seguir, utiliza-se o comando *relplot()* da biblioteca *Seaborn* para se plotar com apenas uma linha de código a relação entre a variável alvo (nota na terceira prova) e uma série de atributos (selecionados por serem bem ranqueados em termos de informação mútua, como se verá adiante nesse trabalho). O argumento *col* permite que se escolha o conjunto de atributos a plotar junto com o atributo alvo. Além disso, fez-se também uso do argumento *hue*, que faz com que se diferencie no gráfico pontos de categorias diferentes.

Figura 10: Relação de vários atributos com a nota da terceira prova de matemática

```
features = ['G2', 'G1', 'absences', 'Walc', 'failures', 'Dalc']
sns.relplot(
    x="value", y="G3", col="variable", col_wrap=3,hue = 'sex',
    data=math.melt(id_vars=["G3",'sex'], value_vars=features), facet_kws=dict(sharex=False),
)
```

```
<seaborn.axisgrid.FacetGrid at 0x6ec41d72e8>
```



A figura acima mostra que, embora com muita variação, há uma leve tendência entre o número de faltas, número de reprovações anteriores e consumo diário de aálocol na nota do aluno. Além disso, embora não seja o objeto de análise desse trabalho, a figura acima sugere que alunos homem apresentam maior consumo de álcool tanto durante os dias de semana quanto durante o fim de semana, o que é observado pela preponderância de pontos do sexo masculino nos níveis mais altos de consumo de álcool. Essa constação será melhor estudada adiante.

No código acima, vale também chamar atenção para um interessante método dos DataFrames, o método *melt()*. O que ele faz é condesar uma série de atributos dependentes em uma única coluna, para se facilitar a plotagem dos dados, conforme mostra a figura abaixo. Nota-se que a coluna “variable” contém vários atributos, e a coluna “value” contém seu valores.

Figura 11: Resultado da utilização do método .melt() dos DataFrames

```
math.melt(id_vars=["G3","sex"], value_vars=features)
```

	G3	sex	variable	value
0	6	F	G2	6
1	6	F	G2	5
2	10	F	G2	8
3	15	F	G2	14
4	10	F	G2	10
...	...	...	...	...
7500	9	M	famrel	5
7501	16	M	famrel	2
7502	7	M	famrel	5
7503	10	M	famrel	4
7504	9	M	famrel	3

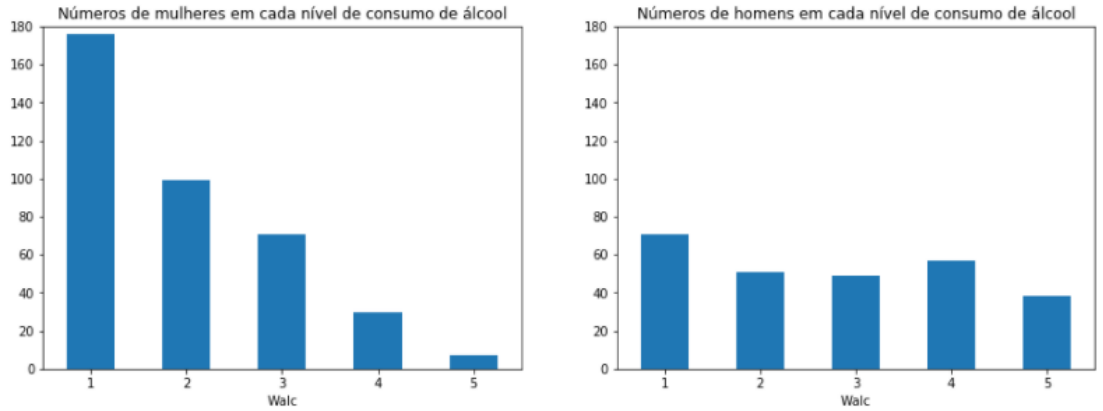
7505 rows × 4 columns

Com relação à diferença no consumo de álcool entre homens e mulheres, a figura a seguir demonstra o perfil de cada sexo quanto a isso.

Figura 12: Diferença no perfil de consumo de álcool entre homens e mulheres

```
fig, axs = plt.subplots(1,2, figsize=(15, 5))
# Plotando a quantidade de alunos de cada sexo por nível de consumo de álcool no FDS
    # Mulheres
port[['Walc','sex']][(port['sex'] == 'F')].groupby(['Walc'])['Walc'].count().plot(kind = 'bar', ax = axs[0])
axs[0].set_xticklabels(labels = ['1','2','3','4','5'], rotation = 0)
axs[0].set_ylim(0,180)
axs[0].set_title('Números de mulheres em cada nível de consumo de álcool')
    # Homens
port[['Walc','sex']][(port['sex'] == 'M')].groupby(['Walc'])['Walc'].count().plot(kind = 'bar', ax = axs[1])
axs[1].set_xticklabels(labels = ['1','2','3','4','5'], rotation = 0)
axs[1].set_ylim(0,180)
axs[1].set_title('Números de homens em cada nível de consumo de álcool')
```

```
Text(0.5, 1.0, 'Números de homens em cada nível de consumo de álcool')
```



O que a figura mostra é que, uma pequena parte das mulheres reporta estar na maior faixa consumo de álcool nos fins de semana. De fato, quanto maior o nível de consumo, menor a quantidade de mulheres o reportam. Já para os homens, há uma quantidade próxima de representantes em cada faixa de consumo.

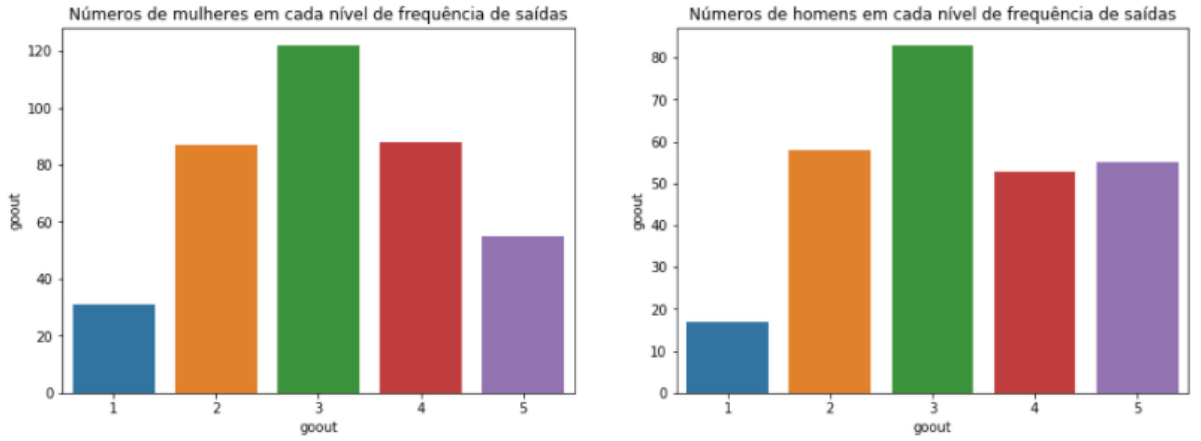
Tentou-se descobrir se há alguma relação entre a frequência reportada de saídas para se divertir e o consumo de álcool, e se essa relação varia entre homens e mulheres. Para se realizar tal investigação, primeiro se buscou entender se há diferença entre o perfil de frequência de saídas entre homens e mulheres. Conforme mostra a figura abaixo, não há. Dessa vez, utilizou-se o comando *barplot()* da biblioteca *Seaborn* para construir os gráficos.

Figura 13: Frequência de saídas de homens e mulheres

```
fig, axs = plt.subplots(1,2, figsize=(15, 5))
# Saídas das mulheres
sns.barplot(x =x.index, y =x['goout'], ax = axs[0])
axs[0].set_title('Números de mulheres em cada nível de frequência de saídas')

# Saídas dos homens
sns.barplot(x = y.index, y = y['goout'], ax = axs[1])
axs[1].set_title('Números de homens em cada nível de frequência de saídas')
```

```
Text(0.5, 1.0, 'Números de homens em cada nível de frequência de saídas')
```



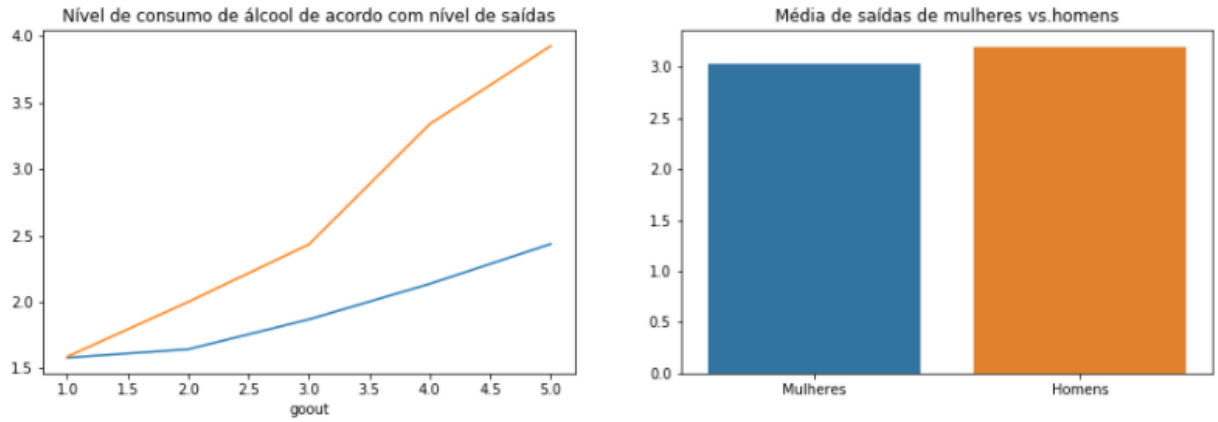
Em seguida, plotou-se como varia o consumo médio de álcool com a quantidade de saídas para cada sexo, conforme mostra a figura abaixo. O que se observa é que o nível de consumo de álcool está sim relacionado à frequência com que a pessoa sái para se divertir, e esse efeito é muito mais pronunciado nos homens, que naturalmente já bebem mais conforme mencionado acima.

Figura 14: Variação do nível de consumo de álcool de acordo com frequência de saídas

```
fig, axs = plt.subplots(1,2, figsize=(15, 4.5))
# Relação álcool x saídas
port[['Walc','goout']][port['sex'] == 'F'].groupby(['goout'])['Walc'].mean().plot(kind = 'line', ax = axs[0],
                                                                                  label = 'Mulheres')
port[['Walc','goout']][port['sex'] == 'M'].groupby(['goout'])['Walc'].mean().plot(kind = 'line', ax = axs[0],
                                                                                  label = 'Homens')
axs[0].set_title('Nível de consumo de álcool de acordo com nível de saídas')

# Homens e mulheres saem na mesma proporção?
sns.barplot(x =['Mulheres','Homens'],
            y = [math['goout'][(math['sex'] == 'F')].mean(),math['goout'][(math['sex'] == 'M')].mean()],
            ax = axs[1])
axs[1].set_title('Média de saídas de mulheres vs.homens')
```

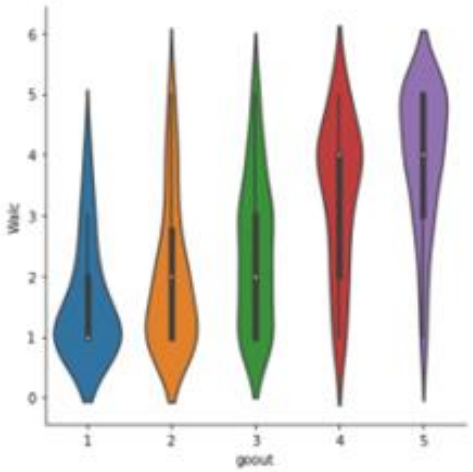
```
Text(0.5, 1.0, 'Média de saídas de mulheres vs.homens')
```



A figura abaixo consiste em um boxplot, mas com formato de violino, construído com o comando *catplot()* da biblioteca *Seaborn* em conjunto com o argumento *kind = 'violin'*, e visa demonstrar de outra forma como varia o consumo de álcool dos homens conforme sua frequência de saídas. Novamente, o que a figura mostra é que os homens que saem mais bebem mais.

Figura 15: Variação da quantidade de álcool consumida em função da frequência de saídas

```
# Diferença no consumo de álcool de acordo com nível de frequência de saídas para homens
sns.catplot(x="goout", y="Walc", data=port[port['sex'] == 'M'], kind="violin");
```



Voltando à análise dos fatores acadêmicos, foi utilizado o comando *lmplot()* da biblioteca *Seaborn* para novamente se investigar a diferença de desempenho entre grupos de alunos diferentes, como, por exemplo, entre os alunos que de sexo diferentes e entre aqueles que frequentaram o maternal ou não.

Primeiramente, plotou-se o desempenho conjunto em português e matemática, mas fazendo a diferenciação entre o sexo dos alunos, utilizando o argumento *hue*. Obteve-se a Figura 16 abaixo, onde também se plotou uma reta (de cor roxa) do que seria um desempenho idêntico nas duas disciplinas. O que se observa é que há uma maior tendência de mulheres terem uma nota maior em português do que em matemática, enquanto que para os homens o desempenho entre as duas disciplinas é mais similar – vide maior proximidade à linha roxa.

Para se confirmar essa hipótese, na Figura 17, foram construídos *boxplots* das notas em cada disciplina para cada sexo. O que se observa é que, de fato, o desempenho apresentado pelos homens em matemática é ligeiramente melhor do que o apresentado pelas mulheres. Já em português, a tendência se inverte, com as mulheres apresentando melhor desempenho.

Figura 16: Desempenho conjunto em português e matemática para cada sexo

```
# Efeito do sexo do aluno no desempenho
sns.lmplot(x="G3_port", y="G3_math", hue="sex", data=notas)
# Plotando uma linha representando um desempenho igual nas duas matérias
plt.plot(range(21), range(21), color='purple', linestyle='dashed', linewidth = 3)
```

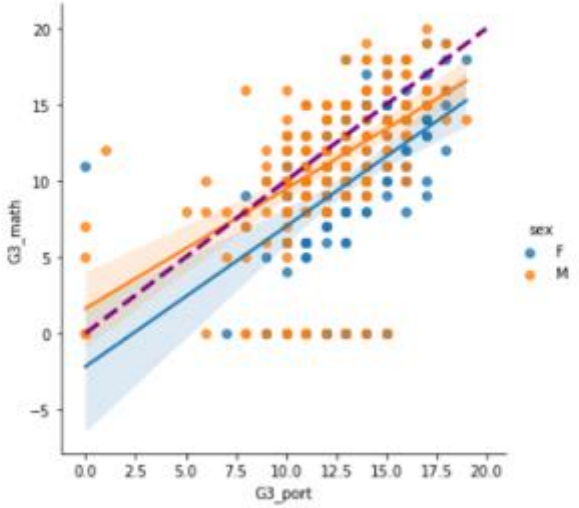
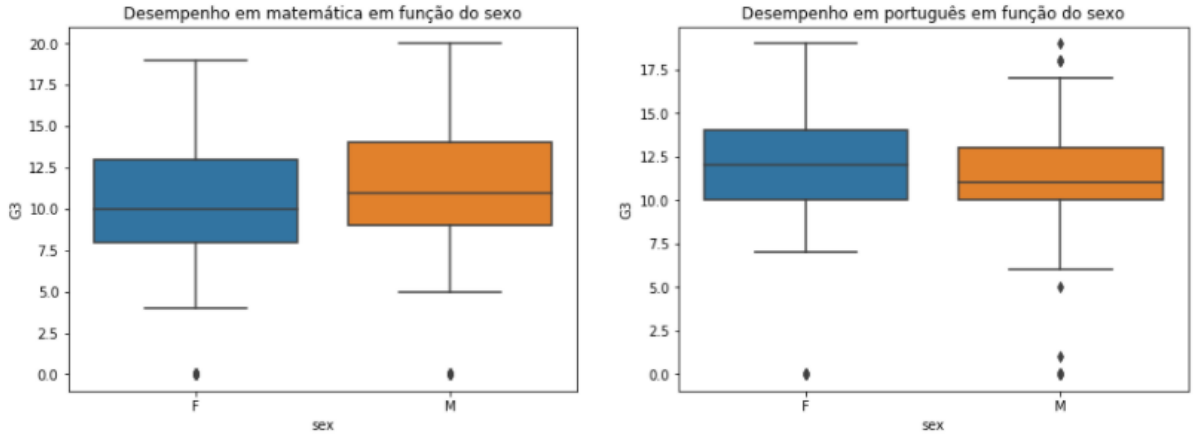


Figura 17: Desempenho de cada sexo em cada disciplina

```
fig, axs = plt.subplots(1,2, figsize=(15, 5))
sns.boxplot(x='sex', y="G3", data=math, ax = axs[0])
axs[0].set_title('Desempenho em matemática em função do sexo')
sns.boxplot(x ='sex', y = 'G3', data = port, ax = axs[1])
axs[1].set_title('Desempenho em português em função do sexo')
```

```
Text(0.5, 1.0, 'Desempenho em português em função do sexo')
```

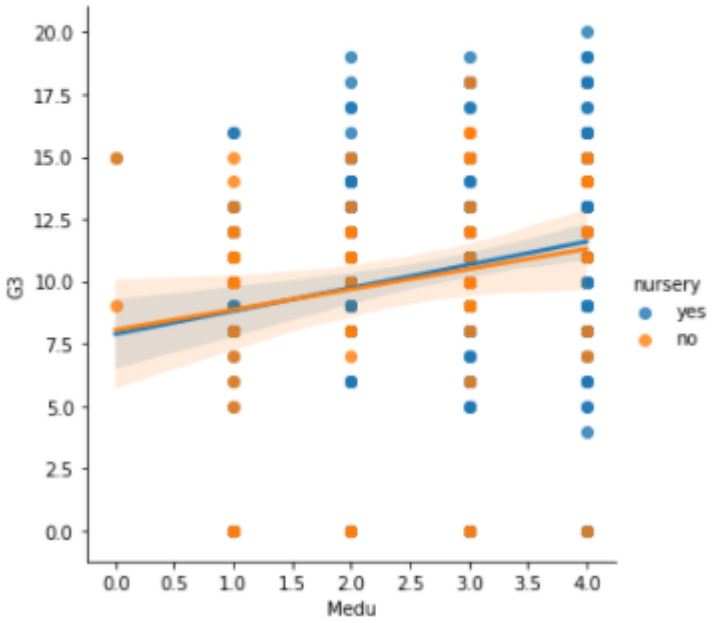


Para se investigar o efeito do aluno ter feito ou não maternal, plotou-se com o método *lmplot()* a nota do aluno em função do grau de escolaridade da mãe, fazendo-se a distinção entre os dois grupos. O que se observa na figura abaixo é que não há distinção significativa entre o desempenho daqueles que cursaram ou não maternal. O gráfico também mostra que em média se melhora ligeramente o desempenho dos alunos com o aumento do grau de escolaridade da mãe. Outra observação que se pode fazer é que à medida que se aumenta o grau de escolaridade da mãe, aumenta a probabilidade dela ter colocado o filho no maternal.

Figura 18: Efeito do grau de escolaridade da mãe do aluno e de se ter feito maternal ou não no desempenho

```
# Efeito do grau da escolaridade da mãe
# e do aluno ter cursado ou não maternal na sua nota
sns.lmplot(x="Medu", y="G3", hue="nursery", data=math)
```

```
<seaborn.axisgrid.FacetGrid at 0x2a7f9f4320>
```



Para se avaliar o efeito do tempo de estudo e da quantidade ingerida de álcool no desempenho acadêmico (desempenho na segunda e terceira prova de português), utilizou-se também o comando *lmplot()*, mas dessa vez com a adição do parâmetro *col*. Esse permite que se faça um gráfico distinto para cada valor do atributo, o que torna mais fácil a visualização quando o atributo tem muitos valores distintos.

Com relação ao tempo de estudo dedicado pelo aluno, o que se vê na Figura 19 é que, embora se possa tirar notas boas estudando pouco, quanto mais o aluno estuda mais improvável fica dele ir mal. Isso pode ser concluído pela redução na quantidade de alunos no quadrante de notas ruins em ambas provas (quadrante inferior-esquerdo), à media em que se aumenta o tempo de estudo.

Já em relação ao consumo de álcool nos fins de semana, o que a Figura 20 mostra é que até um nível de consumo igual a 3, não há significativa diferença. Porém para níveis maiores que 3, se tem uma redução no desempenho.

Figura 19: Efeito do tempo de estudo no desempenho em portugu

```
# Efeito do tempo de estudo no desempenho em portugues
port = pd.read_csv('student-por.csv')
sns.lmplot(
    x='G2', y="G3", hue="studytime", col="studytime",
    data=port, scatter_kws={"edgecolor": 'w'}, col_wrap=3, height=4,
)
```

```
<seaborn.axisgrid.FacetGrid at 0x2a76bfd390>
```

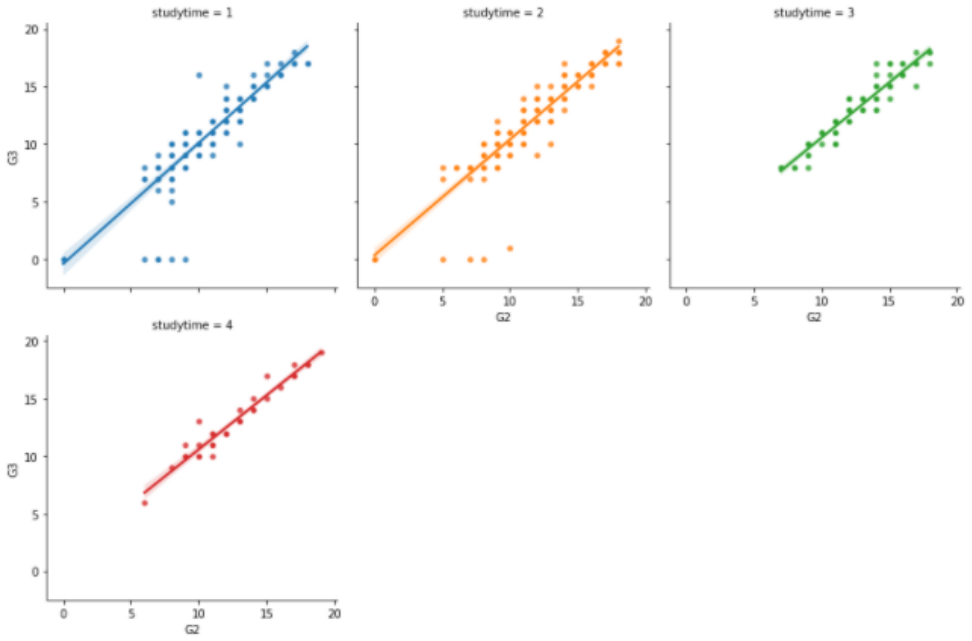
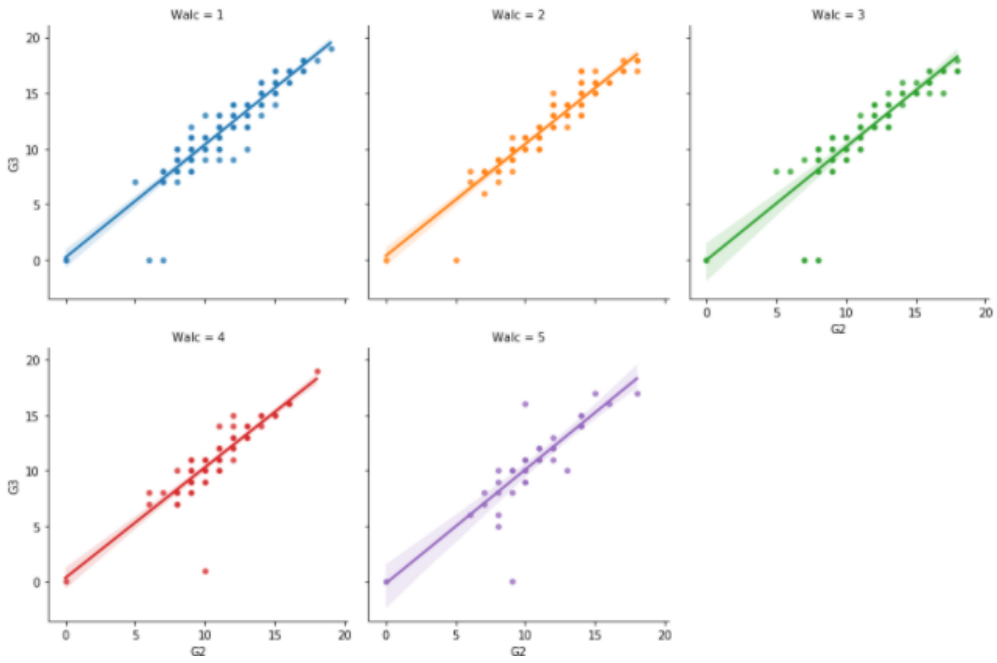


Figura 20: Efeito do nível de consumo de álcool no fim de semana no desempenho em português

```
# Efeito da quantidade de álcool ingerida no fim de semana no desempenho em português
sns.lmplot(
    x='G2', y="G3", hue="Walc", col="Walc",
    data=port, scatter_kws={"edgecolor": 'w'}, col_wrap=3, height=4,
)
```

```
<seaborn.axisgrid.FacetGrid at 0x2a060acf28>
```



Por fim, para se ter uma visão geral do impacto dos atributos na nota final dos alunos (dessa vez foi utilizada a nota em matemática), se criou uma série de boxplots agrupando os atributos em três grupos relacionados à: (1) vida acadêmica; (2) vida familiar; e (3) aspectos sócias, nas Figuras 21, 22, e 23, respectivamente.

Alguns insights que podem ser extraídos das figuras abaixo são os seguintes:

- Dentre os grupos de atributo, o das variáveis acadêmicas é o que mais afeta o desempenho.
- Não há significativa diferença no desempenho de alunos que pagaram aula particular e cursaram maternal. Mas as demais variáveis acadêmicas possuem influência no desempenho acadêmico, em especial a quantidade de reprovações anteriores que está inversamente relacionada ao desempenho.
- Com exceção dos graus de instrução da mãe e do pai, as variáveis familiares possuem uma baixa influência no desempenho.
- As variáveis sociais, de maneira geral, também não possuem um bom poder discriminatório quanto ao desempenho acadêmico.
- Chama-se atenção para o atributo frequência de saídas, que demonstra que se tem uma espécie de ponto ótimo no nível 2. Ou seja, sair pode contribuir um pouco com o desempenho acadêmico, mas sair em excesso passa a ser prejudicial.

Figura 21: Efeito de aspectos relacionados à vida acadêmica no desempenho em matemática

```
estudo = ['paid','schoolsup','nursery','studytime','failures','higher']
fig, axs = plt.subplots(3, 2, figsize=(15, 10)); indice = 0
for i in range(3):
    for j in range(2):
        sns.boxplot(x=estudo[indice], y="G3", data=math, ax = axs[i][j])
        indice = indice + 1
```

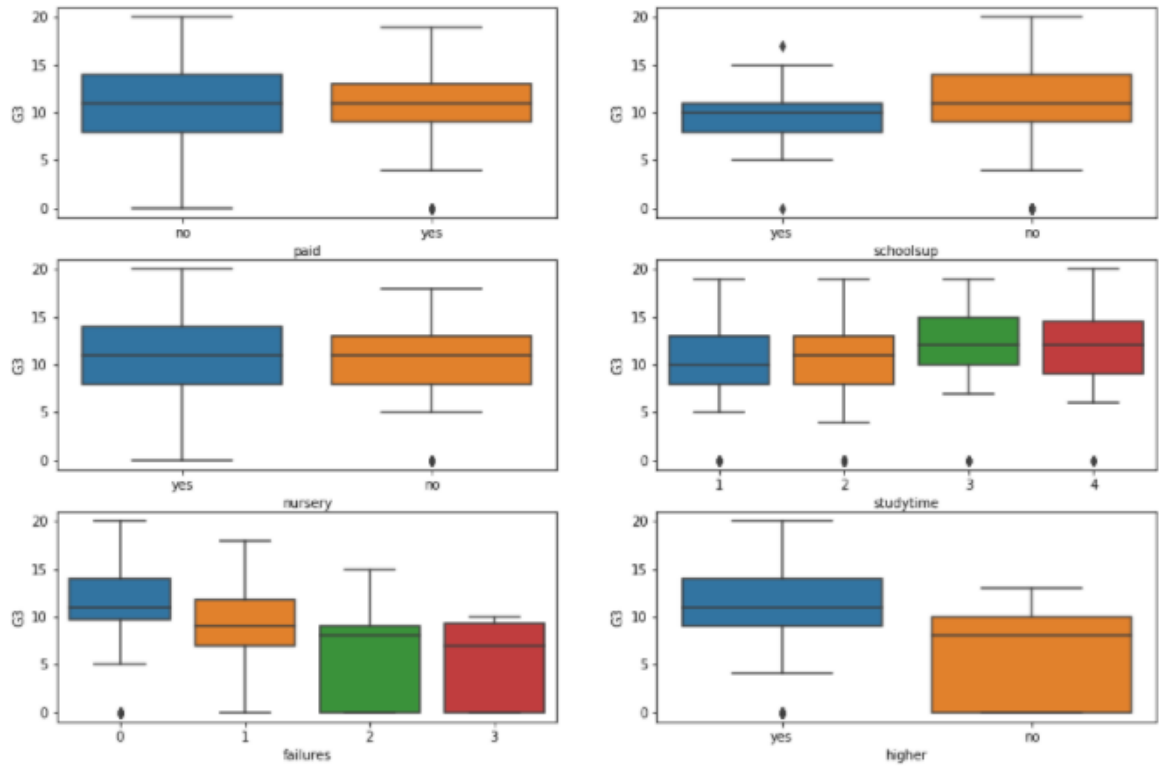


Figura 22: Efeito de aspectos relacionados à vida acadêmica no desempenho em matemática

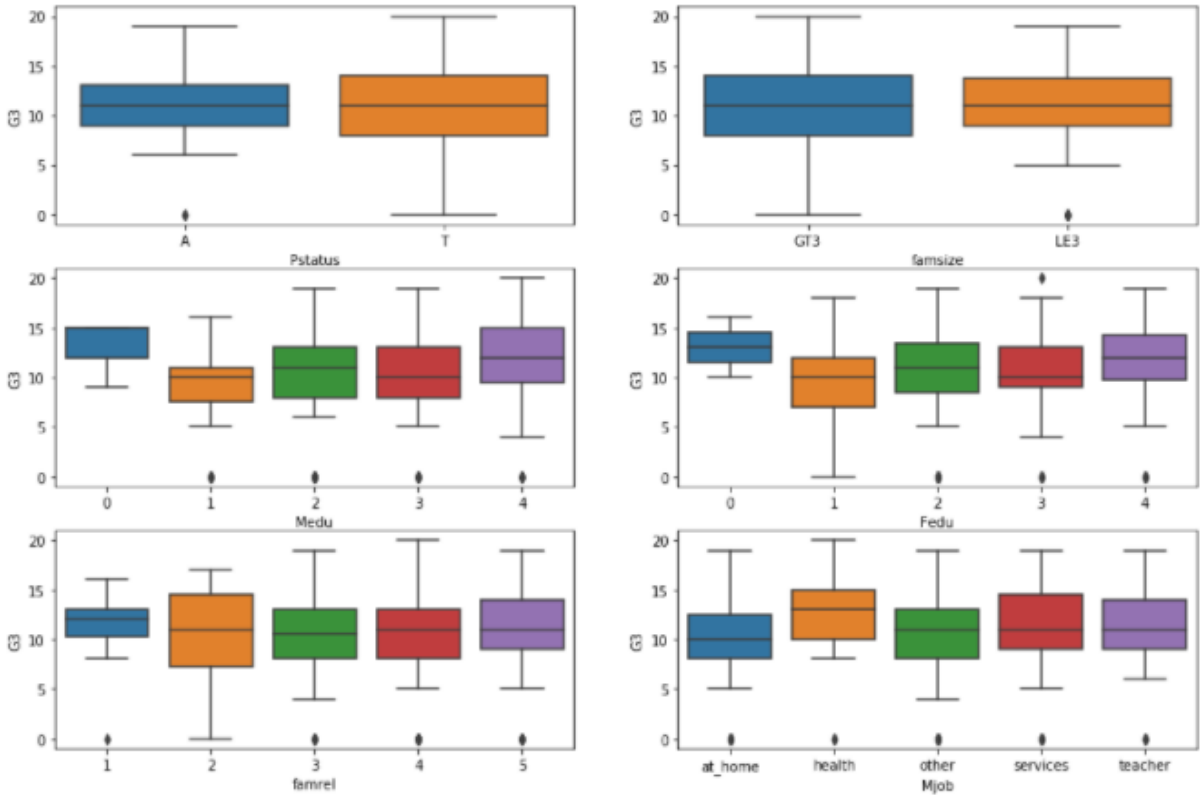
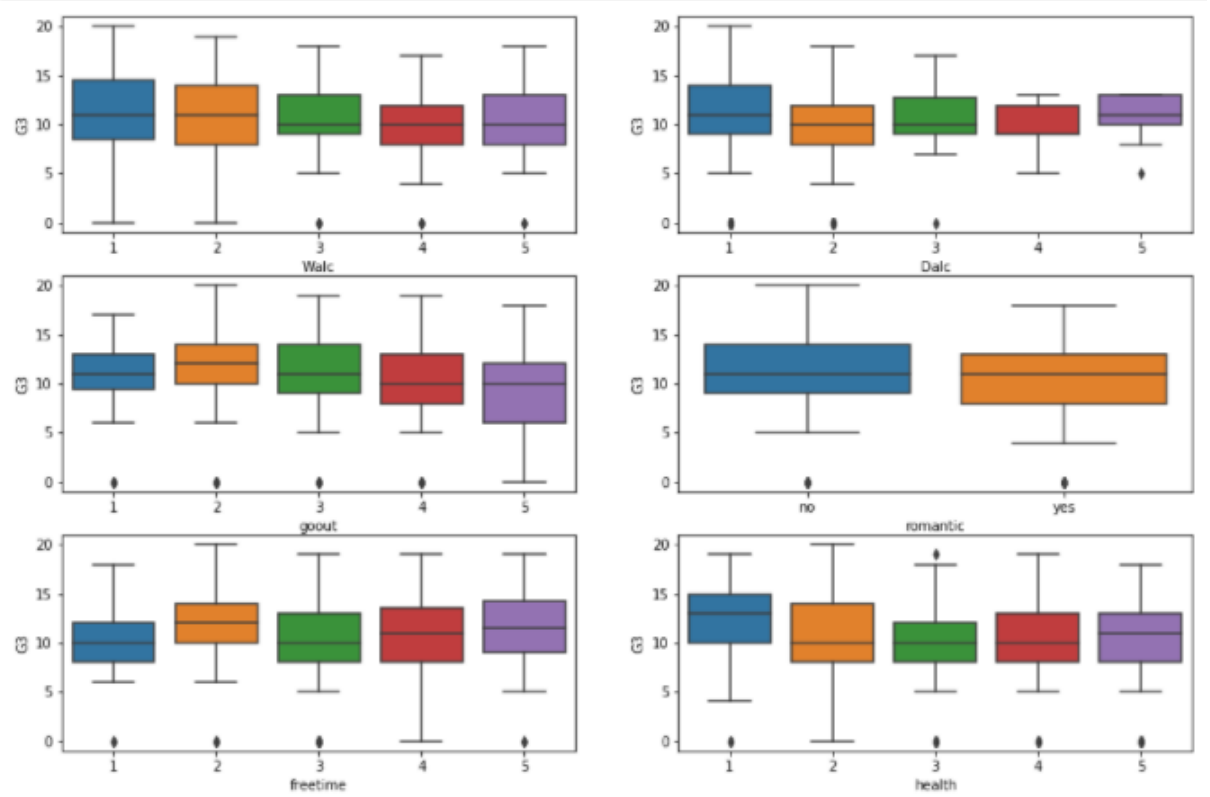


Figura 23: Efeito de aspectos sociais no desempenho em matemática



5. Engenharia de Atributos

5.1. Considerações Iniciais

A Engenharia de Atributos tem por objetivo trabalhar os atributos de forma que possam fornecer informações mais relevantes para o entendimento de um problema. Esse ganho na capacidade do atributo em fornecer informações pode se traduzir em melhores acurácias dos modelos preditivos de aprendizado de máquina em que serão utilizados.

Nesse trabalho se focará em três técnicas de Engenharia de Atributos: (1) clusterização; (2) criação de atributos derivados ou sintéticos; (3) levantamento das informações mútuas de cada atributo e suas correlações.

5.2. Clusterização dos alunos

Clusterizarização, no contexto da Ciência de Dados, é uma técnica que visa agrupar registros de acordo com semelhanças entre atributos. No caso do presente trabalho, essa técnica será utilizada para agrupar os alunos de acordo com semelhanças em relação a aspectos: (1) relacionados à vida academia, (2)

familiares; (3) sociais/comportamentais. Além disso, também será feita uma quarta clusterização, essa mais geral, reunindo todos atributos das outras três, de forma procurar semelhanças como um todo entre os alunos.

O que se visa é que reunindo o aluno em grupos, o grupo forneça uma informação mais relevante para a previsão do desempenho acadêmico. Os clusteres serão formados com base nos atributos abaixo:

- Cluster com base em fatores acadêmicos: se o aluno pagou ou não aula particular, se ele faz ou não aula particular, se frequentou maternal, nível de estudo, quantidade de reprovações anteriores, faltas no ano letivo (essa variável será normalizada, para se reduzir o impacto de poucos valores excepcionalmente altos).
- Cluster com base em fatores familiares: tamanho da família, nível de educação da mãe, nível de educação do pai, nota da relação familiar, se a família recebe algum apoio da escola, se os pais vivem juntos.
- Cluster com base em fatores sociais/comportamentais: consumo de álcool no fim de semana, consumo de álcool em dias de semana, frequência de saídas para diversão, se o aluno encontra-se em um relacionamento, tempo livre, nota dada à saúde.

O método utilizado para fazer a clusterização será o *k-means* que, basicamente agrupa os alunos de acordo com sua proximidade no espaço vetorial formado pelos valores de cada atributo. Foi utilizado o objeto *Kmeans* da módulo *cluster* da biblioteca *Scikit Learn*.

O código demonstrado na figura abaixo foi utilizado para se produzir os clusteres. O código inteiro utilizado pode ser encontrado no repositório citado no fim deste trabalho no *Jupyter Notebook* – Clusterização dos alunos. O dataset utilizado foi o das notas de matemática pre-processadas para se converter os valores categóricos em numéricos, conforme demonstrado anteriormente nesse trabalho. Chama-se atenção também que para o atributo das faltas do aluno no ano letivo, foi feita uma normalização, pois alguns poucos alunos faltaram muito, e isso poderia reduzir a eficiência da clusterização. Por fim, após alguns testes, foi concluído que a quantidade mais adequada de clusteres é de dois.

Figura 24: Procedimento para Clusterização dos Alunos

```
mathModificada['absences_normalized'] =(mathModificada['absences'] -
                                        mathModificada['absences'].mean(axis=0)) /
                                        mathModificada['absences'].std(axis=0)
```

```
# Agrupando fatores relacionados
estudo = ['paid','activities','nursery','studytime','failures','absences_normalized']
familiares = ['famsize','Medu','Fedu','famrel','famsup','Pstatus']
sociais = ['Walc','Dalc','goout','romantic','freetime','health']
geral = ['paid','activities','nursery','studytime','failures','absences_normalized',
                'Walc','Dalc','goout','romantic','freetime','health']
geral = ['failures','absences','Mjob','reason','schoolsup','paid']
clusters = ["Cluster_estudo","Cluster_familiares","Cluster_sociais",'Cluster_geral']
```

```
# Criando os clusters de alunos baseado em cada conjunto de características
from sklearn.cluster import KMeans
n_clusters = 2
kmeans = KMeans(n_clusters = n_clusters, n_init = 10, random_state=0)
mathModificada["Cluster_estudo"] = kmeans.fit_predict(mathModificada[estudo])
mathModificada["Cluster_familiares"] = kmeans.fit_predict(mathModificada[familiares])
mathModificada["Cluster_sociais"] = kmeans.fit_predict(mathModificada[sociais])
mathModificada['Cluster_geral'] = kmeans.fit_predict(mathModificada[geral])
```

A figura a seguir demonstra um trecho do DataFrame resultante agora com os atributos de cada *cluster*.

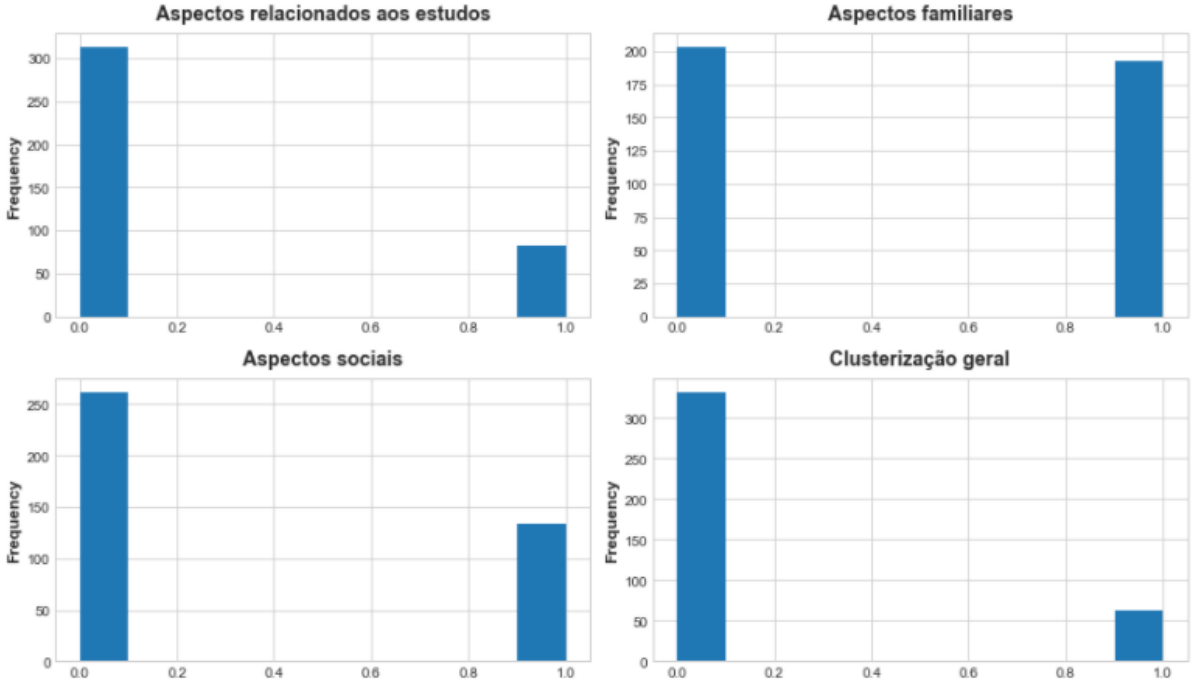
Figura 25: Trecho do DataFrame resultante filtrando as colunas de clusters

```
mathModificada[clusters].head()
```

	Cluster_estudo	Cluster_familiares	Cluster_sociais	Cluster_geral
0	0	1	0	0
1	0	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	1	0	0

A figura a seguir demonstra a quantidade de alunos pertencem a cada um dos clusteres criados. Na sequência, se procurará entender o perfil de aluno que cada cluster reuniu.

Figura 26: Distribuição dos alunos em cada um dos clusteres



As Figuras 27, 28 e 29 a seguir visam elucidar o perfil de aluno que cada cluster representa. Isso é feito se construído *boxplots* violino para cada atributo usado na construção do *cluster*.

Com relação ao agrupamento dos alunos em termos de aspectos acadêmicos/estudantis, nota-se que se formou grupos predominantemente antagônicos em relação a quanto ao tempo dedicado ao estudo, faltas e reprovações anteriores. Os clusteres não demonstraram diferença significativa em relação aos outros atributos.

Já no agrupamento por fatores familiares, nota-se que a distinção se deu primordialmente com base na educação dos pais dos alunos. Ou seja se formou um grupo reunindo os alunos com pais com um maior grau de escolaridade e outro contendo alunos pais com um menor grau de escolaridade.

Por fim, o agrupamento por aspetos sociais e comportamentais, reuniu alunos antagonicamente em relação a consumo de álcool num maior grau, e em relação a tempo livre e frequência de saídas num menor grau.

Figura 27: Características reunidas no cluster de aspectos acadêmicos

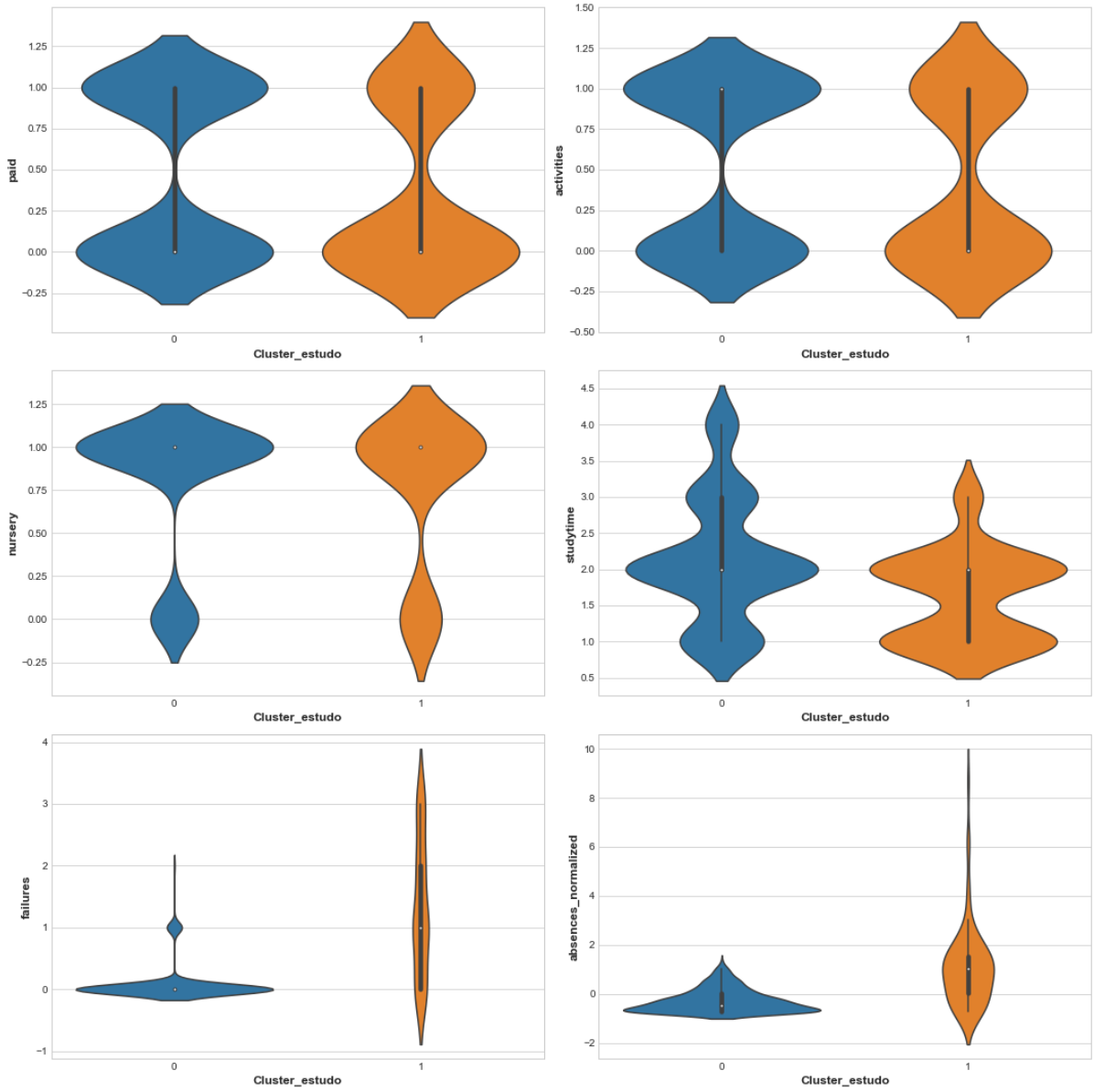


Figura 28: Características reunidas no cluster de aspectos familiares

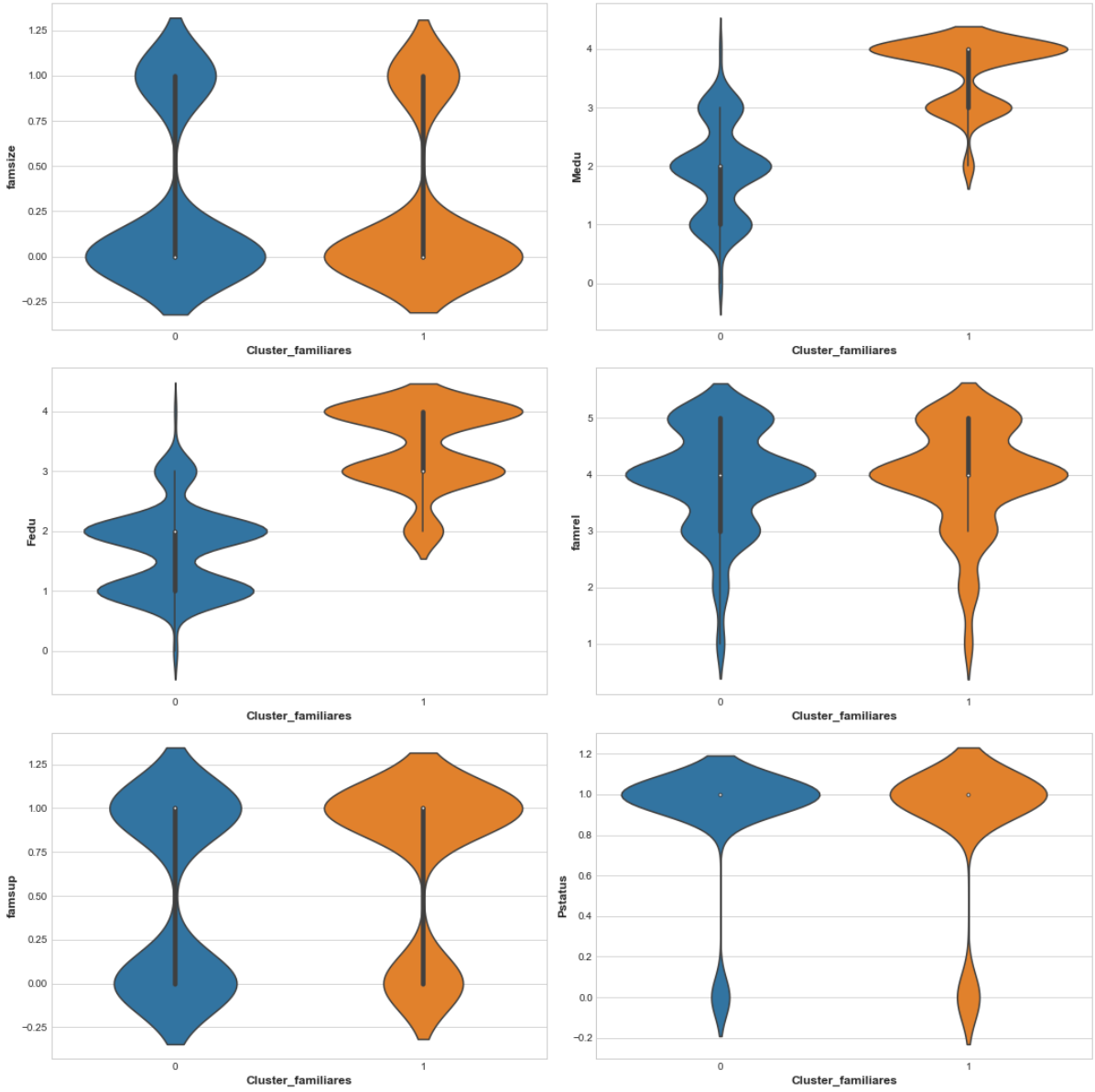
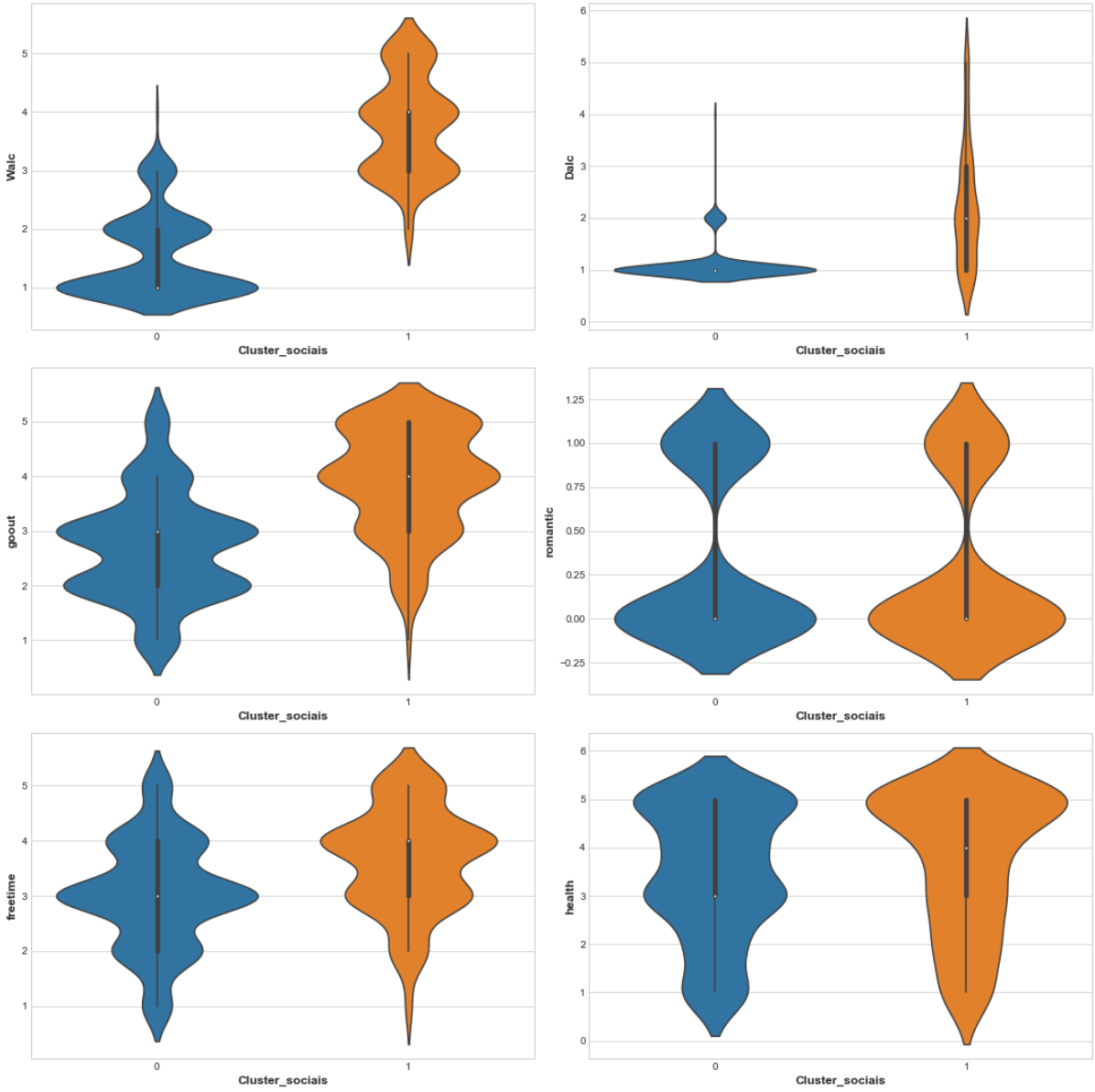
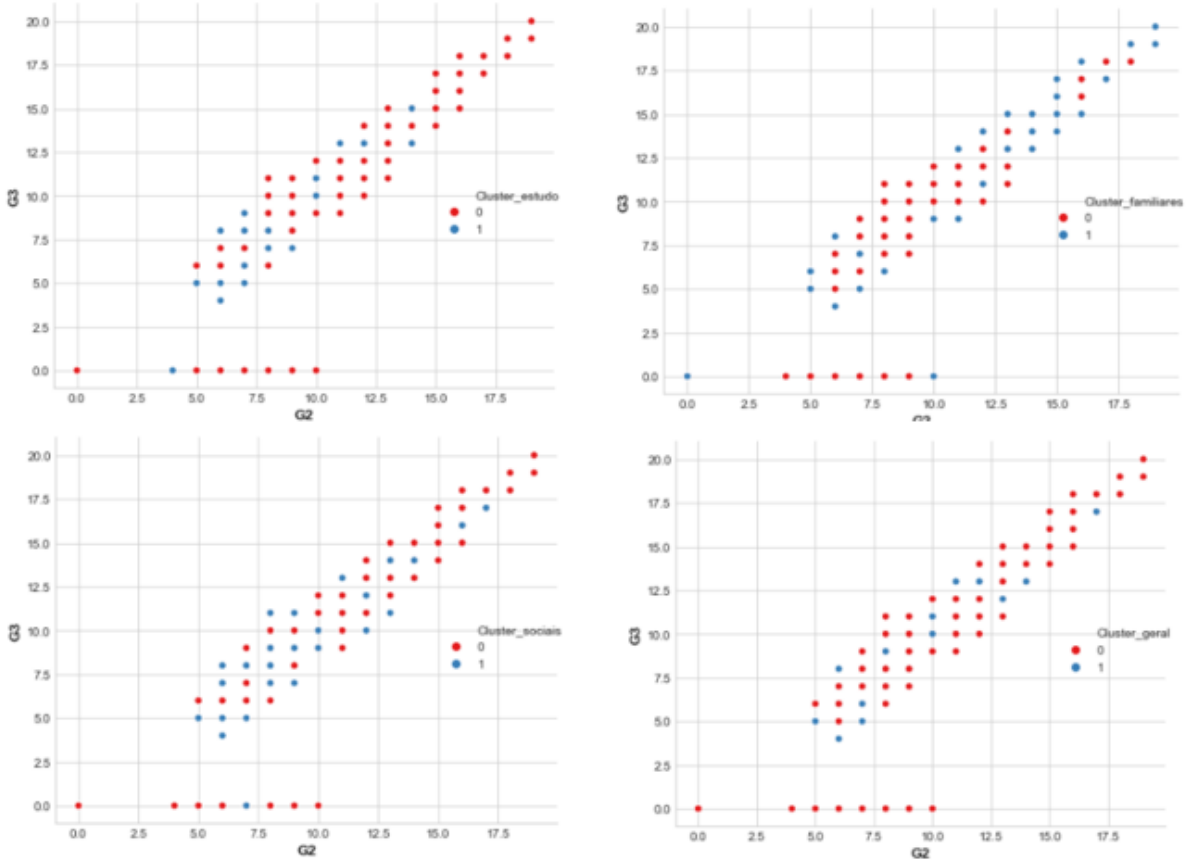


Figura 29: Características reunidas no cluster de aspectos sociais/comportamentais



A figura a seguir visa apresentar uma avaliação do impacto que cada agrupamento teve na capacidade de se prever o desempenho acadêmico dos alunos. Basicamente, se plota o desempenho na segunda e terceira prova de matemática com o comando *relplot()* da biblioteca *Seaborn*, fazendo-se a distinção entre os clusteres com o parâmetro *hue*. Nota-se que o agrupamento em termos de fatores acadêmicos foi o que demonstrou mais valia em termos de distinguir o desempenho acadêmico, em especial dos alunos que foram bem. O agrupamento por fatores familiares também demonstrou ter alguma valia na previsão dos casos de melhor desempenho.

Figura 30: Impacto dos clusteres na previsibilidade do desempenho acadêmico



5.3. Construção de atributos derivados/sintéticos

A construção de novos atributos a partir dos atributos originais tem também o intuito de condensar a informação de tal maneira que se tenha maior poder de prever o alvo em questão. Trata-se de uma etapa que pode requerer conhecimento de domínio, ou seja, conhecimento do assunto da análise, e também uma certa dose de criatividade.

Para o trabalho atual, decidiu-se criar três atributos sintéticos (cuja valia em termos de prever o desempenho acadêmico será avaliada na próxima seção), são eles:

- Razão do tempo que o aluno tem livre para o tempo de estudo.
- Razão entre a frequência de saídas do aluno e seu tempo de estudo.
- Um somatório das notas dadas pelo aluno a fatores que, à primeira vista, poderiam contribuir com seu desempenho.

A figura a seguir demonstra o código utilizado para a criação de cada um desses atributos.

Figura 31: Criação dos atributos sintéticos

```
# Criando o atributo razão tempo livre e tempo de estudo
mathModificada['RazaoLivreEstudo'] = mathModificada['freetime']/mathModificada['studytime']
# Criando o atributo razão tempo de saídas e tempo de estudo
mathModificada['RazaoSaidasEstudo'] = mathModificada['goout']/mathModificada['studytime']
# Criando o atributo com somatório de coisas boas
coisasBoas = ['famrel','Medu','Fedu','studytime','freetime','activities','paid']
mathModificada['Somatorio'] = mathModificada[coisasBoas].sum(axis = 1)
```

```
# Conferindo o atributo "Somatorio"
mathModificada[['famrel','Medu','Fedu','studytime','freetime','activities','paid','Somatorio']].head()
```

	famrel	Medu	Fedu	studytime	freetime	activities	paid	Somatorio
0	4	4	4	2	3	0	0	17
1	5	1	1	2	3	0	0	12
2	4	1	1	2	3	0	1	12
3	3	4	2	3	2	1	1	16
4	4	3	3	2	3	0	1	16

5.4. Infomação mútua e correlação entre atributos

A Informação Mútua é uma das métricas que mede a capacidade de um atributo em permitir a previsão de determinado alvo. De maneira mais técnica, dentro da Teoria da Informação, a métrica de Informação Mútua entre duas variáveis, basicamente, mede o quanto se reduz a incerteza sobre o valor de uma variável conhecendo-se o valor da outra.

Para se obter os valores de Informação Mútua, utilizou-se o objeto *mutual_info_regression* do módulo *feature_selection* da biblioteca *Sckit Learn*. Utilizou-se o dataset sobre o desempenho em matemática, ao qual foi incluídos os atributos de clusterização, bem como os atributos sintéticos gerados anteriormente. Vale destacar também que o código completo utilizado nessa etapa se encontrado no repositório mencionado no fim desse trabalho no *Jupyter Notebook* – Informação Mútua e Correlação.

O código utilizado para a geração dos valores de informação mútua está mostrado na figura abaixo.

Figura 32: Código utilizado para geração dos valores de informação mútua

```
from sklearn.feature_selection import mutual_info_regression
```

```
# Carregando a tabela gerada no notebook "Clusterização dos alunos"
mathClusters = pd.read_excel('Math modificada com clusters.xlsx')
clusters = ["Cluster_estudo","Cluster_familiares","Cluster_sociais",'Cluster_geral']

# Anexando os atributos de clusteres no dataset mathModificada
mathModificada = pd.concat([mathModificada,mathClusters[clusters]],axis = 1)
```

```
# Separando a variável "alvo"
y = mathModificada.pop('G3')
```

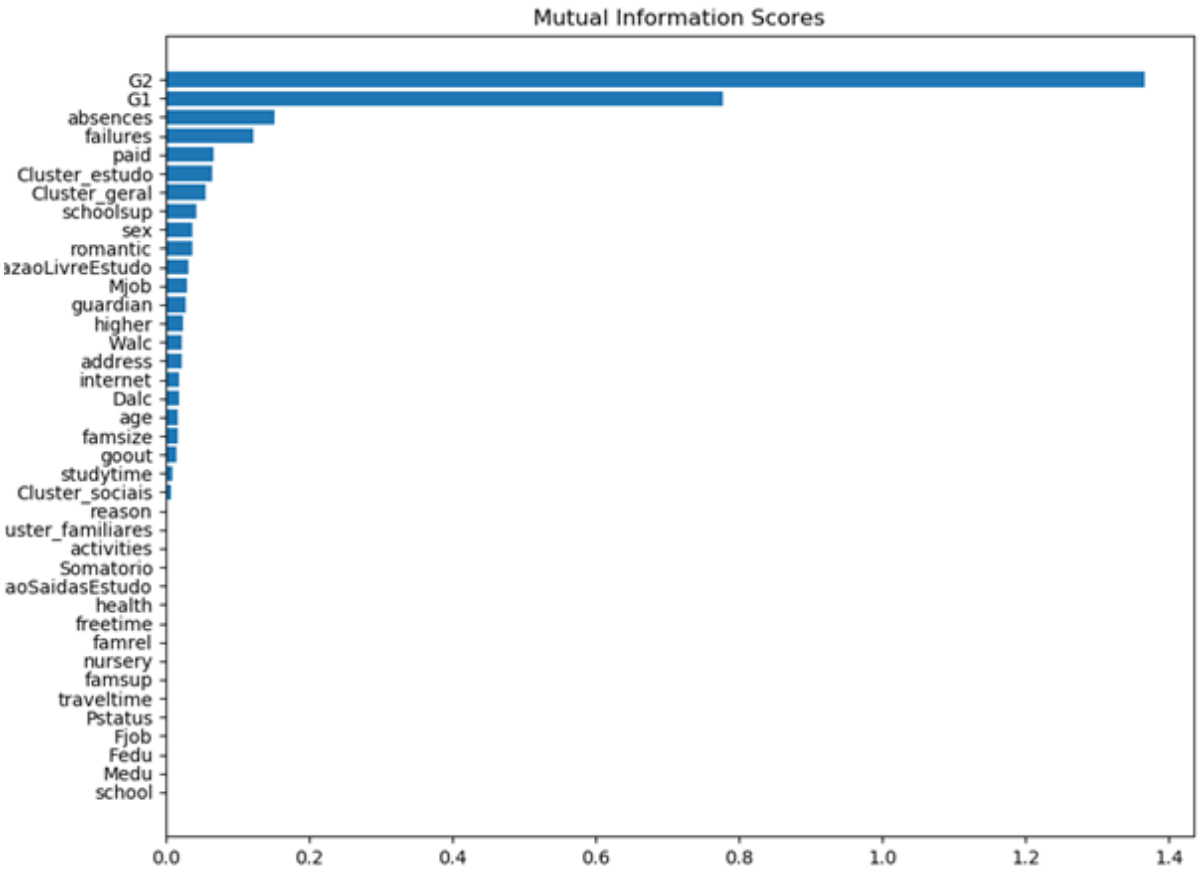
```
# A função abaixo, primeiro, obtém os valores de mutual information e
# na sequência cria uma lista associando-os ao seu atributo.
def make_mi_scores(X, y, discrete_features):
    mi_scores = mutual_info_regression(X, y, discrete_features=discrete_features)
    mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)
    mi_scores = mi_scores.sort_values(ascending=False)
    return mi_scores
```

```
mi_scores = make_mi_scores(mathModificada, y, True )
```

Fonte: A função make_mi_scores acima foi adaptada a partir do código apresentado no curso disponibilizado na plataforma Kaggle (site: https://www.kaggle.com/ryanholbrook/mutual-information)

Os resultados obtidos estão mostrados na figura abaixo.

Figura 33: Informação mútua de cada atributo



Como era de se esperar as notas das primeiras e segunda prova possuem a maior informação mútua para se prever a nota da terceira prova, com os atributos número de faltas e reprovações anteriores aparecendo na sequência.

Interessante também notar como que os agrupamentos à base das características estudantis e sociais ficaram bem ranqueados. Além disso, o atributo sintético que faz a razão entre tempo de saídas e tempo de estudo ficou melhor posicionado do que os atributos utilizados em sua criação. Isso demonstra a importância de se utilizar as técnicas de engenharia de atributos.

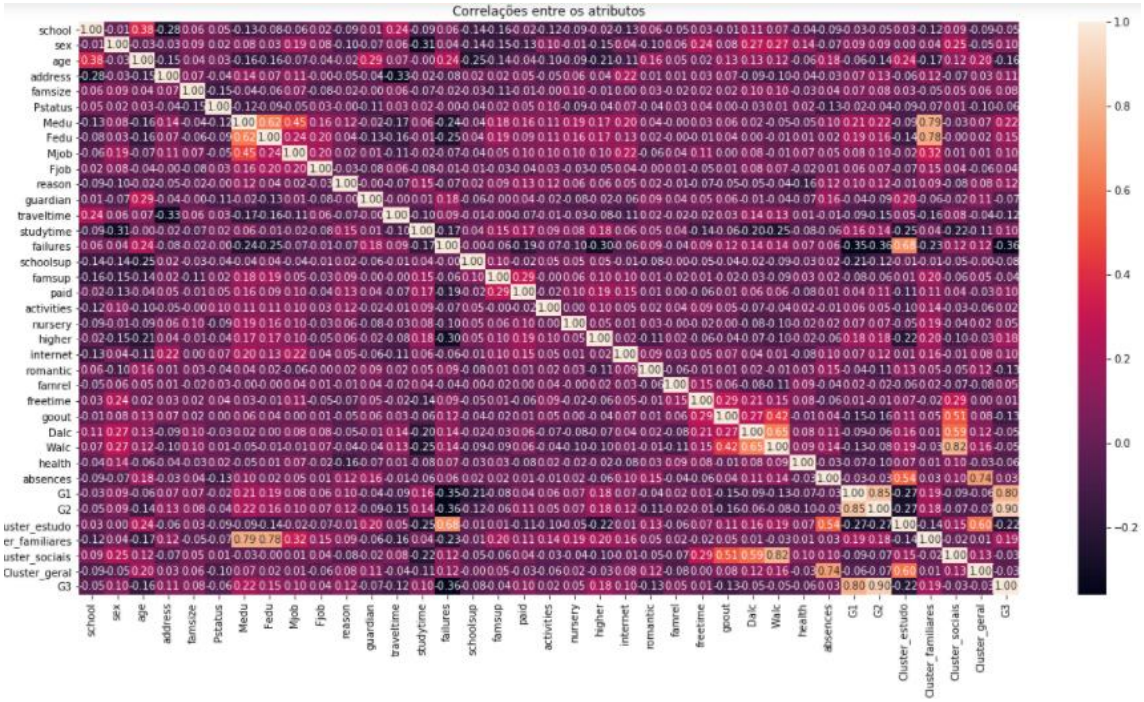
Já quanto a correlação entre os atributos a classe *DataFrames* da biblioteca *Pandas* já apresenta um método extremamente prático para obtê-las, o *.corr()*. Esse método gera uma matriz, que pode ser plotada na forma de um mapa de calor com o comando *heatmap()* da biblioteca *Seaborn*. Abaixo se demonstra o código utilizado, bem como o mapa de calor construído.

Figura 34: Código utilizado para obter as correlações entre os atributos e expressá-las na forma de mapa de calor

```
# Eliminando os atributos sintéticos
mathModificada = mathModificada.drop(['Somatorio','RazaoLivreEstudo','RazaoSaidasEstudo'], axis = 1)
# Obtendo a matriz de correlações
mathModificada = pd.concat([mathModificada, math['G3']], axis = 1)
correlacoes = mathModificada.corr()

# Plotando a matriz
plt.figure(figsize=(20,10))
sns.heatmap(correlacoes,annot = True, fmt = ".2f")
plt.title("Correlações entre os atributos")
plt.show()
```

Figura 35: Mapa de calor das correlações entre atributos

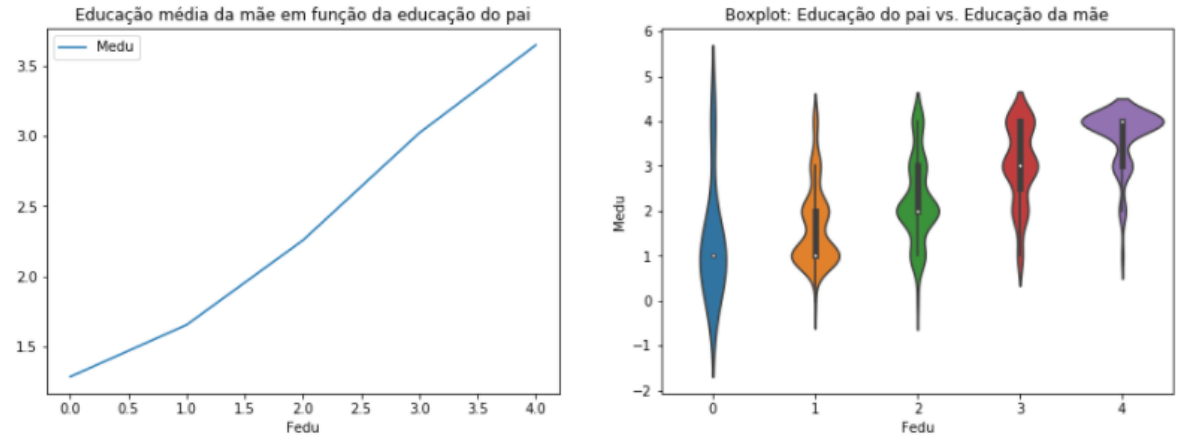


O mapa de correlações aponta muitas correlaçoes interessantes. Uma que chama a atenção é a correlação entre o grau de escolaridade dos pais dos alunos. Aparentemente, há uma certa tendência dos casais se formarem entre pessoas da mesma faixa de escolaridade. Para melhor investigar isso, foi construída as figuras abaixo plotando a média de escolaridade dos pais vs a das mães tanto e um *boxplot* associando as duas variáveis. Como se pode ver, de fato a tendência existe.

Figura 36: Grau de escolaridade da mãe vs. a do pai

```
fig, axs = plt.subplots(1,2, figsize=(15, 5))
# Gráfico de linha
port[['Fedu','Medu']].groupby('Fedu').mean().plot(kind = 'line', ax = axs[0])
axs[0].set_title('Educação média da mãe em função da educação do pai')
# Boxplot
sns.violinplot(x="Fedu", y="Medu", data=port, ax = axs[1]);
axs[1].set_title('Boxplot: Educação do pai vs. Educação da mãe')
```

```
Text(0.5, 1.0, 'Boxplot: Educação do pai vs. Educação da mãe')
```



6. Criação de Modelos de Machine Learning

6.1. Considerações iniciais

Nesta etapa se mostrará o resultado da aplicação de um modelo classificador – Árvore de Decisão – para se fazer dois tipos de previsão para a disciplina de matemática: (1) se o aluno passou ou não (tirou uma nota maior ou igual a 10 na última/terceira prova), e (2) a nota que o aluno tirou na terceira prova. Utilizou-se dois conjuntos de atributos, um formado pelos 10 atributos melhor ranqueados quanto a Informação Mútua, e outro contendo esses mesmos 10 atributos mais as notas na primeira e segunda prova de matemática.

Em cada um dos casos, foi conduzida uma pesquisa empírica com relação ao valor dos hiperparâmetros a serem utilizados. Na sequência se avaliou a acurácia do modelos, utilizando a métrica de acurácia para a classificação binária (qual foi a

proporção de acerto do modelo) e a métrica de erro absoluto médio para a avaliação da previsão da nota do aluno.

Para o caso da classificação binária (se o aluno passou não passou), também se plotou a matriz de confusão para cada um dos modelos.

A figura abaixo demonstra os conjuntos de atributos criados, bem como os principais os módulos da biblioteca *Sckit Learn* utilizados.

Figura 37: Definição dos conjuntos de atributos e importação das bibliotecas necessárias

```
# Atributos melhor ranqueados em termos de informação mútua,
#porém sem as notas das duas primeiras provas.
X = atributos[informacaoMutua.index.to_list()[2:12]]
# Atributos com as notas das duas primeiras provas
XcomNotas = atributos[informacaoMutua.index.to_list()[:12]]
```

```
# Importando os módulos aplicáveis
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

Importante mencionar também que se trabalhou com o *dataset* de matemática em que todos valores categóricos foram convertidos para valores numéricos e ao qual foram adicionados os atributos sintéticos bem como os agrupamentos criados.

Todo o código da análise aqui demonstrada encontra-se no repositório mencionado no fim desse trabalho no *Jupyter Notebook* – Modelos de Machine Learning para Previsão de Notas.

6.2. Usando o modelo Árvore de Decisão para prever se o aluno passou

A primeira etapa constituiu-se de criar o atributo alvo – se o aluno foi aprovado ou reprovado – com base na nota da última nota. O código para se fazer isso está mostrado na figura a seguir.

Figura 38: Criação da variável alvo binária

```
alvo = pd.read_csv('student-mat.csv', usecols = ['G3'])
```

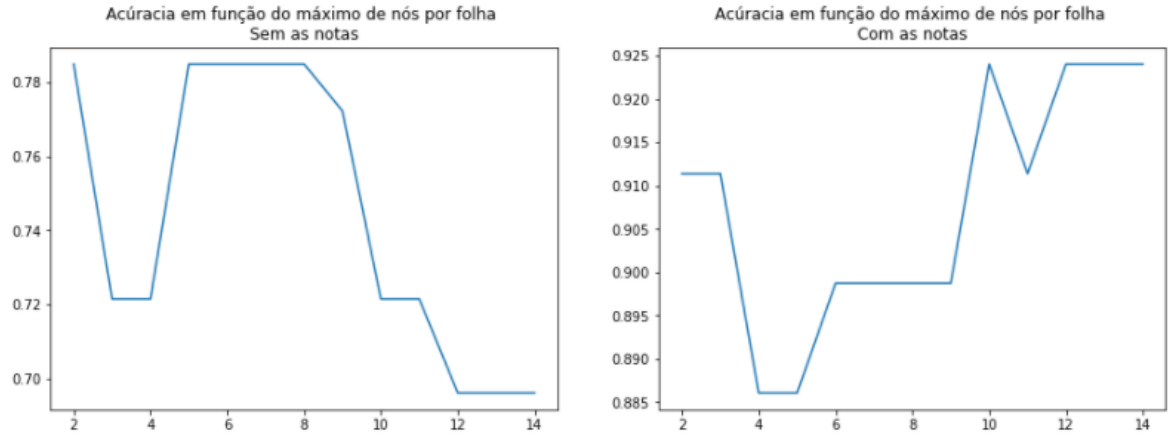
```
alvo['Situacao'] = 0
for i in range(len(alvo['G3'])):
    nota = alvo['G3'][i]
    if nota >= 10:
        alvo['Situacao'][i] = 1 # "1" representará o aluno aprovado
    else:
        alvo['Situacao'][i] = 0 # Representará o aluno reprovado
```

Em seguida, avaliou-se a variação da acurácia do método variando-se o hiperparâmetro *max_leaf_nodes*. Fez-se isso para ambos conjuntos de atributos (com e sem as notas das primeiras provas). Vale mencionar, que se utilizou uma proporção de 80% dos dados para treinamento do modelo, e 20% para teste/validação, isso é facilmente implementado utilizando-se o comando *train_test_split()* da biblioteca *Scikit Learn*. Abaixo, encontra-se o código utilizado nessa etapa, bem como os resultados de acurácia para cada conjunto de atributos, lembrando que como se trata de uma classificação binária, achou-se por bem usar acurácia do modelo como métrica de desempenho.

Figura 39: Código utilzado para se avaliar a acurácia do modelo árvore de decisão se variando o hiperparâmetro max_leaf_nodes, para cada conjunto de atributos

```
fig, axs = plt.subplots(1, 2, figsize=(15, 5))
atributosTeste = [X, XcomNotas]
labels = ['Sem as notas','Com as notas']
i = 0
for atributo in atributosTeste:
    # Criando subsets para treino e validação do modelo
    treino_X, val_X, treino_notas, val_notas = train_test_split(atributo, alvo['Situacao'], test_size = 0.2, random_state = 1)
    # Será testado o efeito do parâmetro max_leaf_nodes na acurácia do modelo
    acuracias = []
    for nodes in range(2,15):
        # Criando o modelo
        notasArvoreDecisao = DecisionTreeRegressor(max_leaf_nodes=nodes,random_state=0)
        # Treinando o modelo
        notasArvoreDecisao.fit(treino_X, treino_notas)
        # Testando o modelo
        previsoes = notasArvoreDecisao.predict(val_X)
    # Obtendo a acurácia do modelo
        acuracias.append(accuracy_score(val_notas, previsoes.round()))
    # Plotando os resultados
    pd.Series(data = acuracias, index = [i for i in range(2,15)]).plot(kind = 'line', ax = axs[i])
    axs[i].set_title('Acúracia em função da profundidade máxima da árvore \n' + labels[i])
    i = i + 1
```

Figura 40: Acurácias para cada conjunto de atributos



Chega-se à conclusão que para os atributos sem as notas um valor ótimo do hiperparâmetro *max_leaf_nodes* é 5, enquanto que para o conjunto de atributos com as notas, esse valor é de 9. Utilizando-se esses valores ótimos agora se avalia a o efeito do hiperparâmetro *min_samples_leaf*. O código, bem como o resultado estão demonstrados abaixo.

Figura 41: Código para se testar o efeito do parâmetro min_samples_leaf para os dois conjuntos de atributos

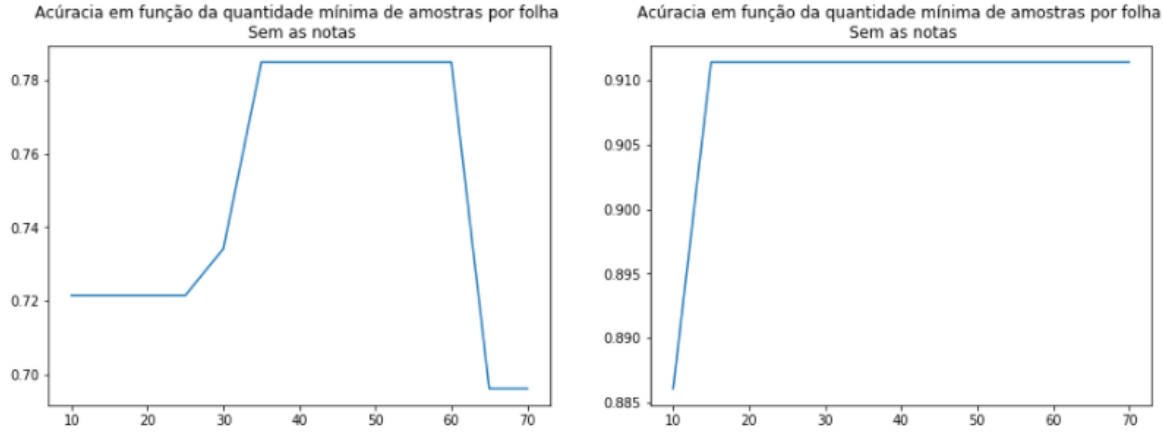
```
# Modelo sem as notas primeira e segunda prova
treino_X, val_X, treino_notas, val_notas = train_test_split(X, alvo['Situacao'], test_size = 0.2, random_state = randint(0,5))
# Será testado o efeito do parâmetro "min_samples_leaf" na acurácia do modelo
acuracias = []
for leafs in range(2,15):
    # Criando o modelo
    notasArvoreDecisao = DecisionTreeRegressor(max_leaf_nodes=5,min_samples_leaf = leafs*5,random_state=randint(0,5))
    # Treinando o modelo
    notasArvoreDecisao.fit(treino_X, treino_notas)
    # Testando o modelo
    previsoes = notasArvoreDecisao.predict(val_X)
    # Obtendo a acurácia do modelo
    acuracias.append(accuracy_score(val_notas, previsoes.round()))

fig, axs = plt.subplots(1, 2, figsize=(15, 5))
pd.Series(data = acuracias, index = [i*5 for i in range(2,15)]).plot(kind = 'line', ax = axs[0])
axs[0].set_title('Acúracia em função da quantidade mínima de amostras por folha \n Sem as notas')

# Modelo com as notas primeira e segunda prova
treino_X, val_X, treino_notas, val_notas = train_test_split(XcomNotas, alvo['Situacao'], test_size = 0.2, random_state = 1)
# Será testado o efeito do parâmetro "min_samples_leaf" na acurácia do modelo
acuracias = []
for leafs in range(2,15):
    # Criando o modelo
    notasArvoreDecisao = DecisionTreeRegressor(max_leaf_nodes= 9,min_samples_leaf = leafs*5,random_state=1)
    # Treinando o modelo
    notasArvoreDecisao.fit(treino_X, treino_notas)
    # Testando o modelo
    previsoes = notasArvoreDecisao.predict(val_X)
    # Obtendo a acurácia do modelo
    acuracias.append(accuracy_score(val_notas, previsoes.round()))

pd.Series(data = acuracias, index = [i*5 for i in range(2,15)]).plot(kind = 'line', ax = axs[1])
axs[1].set_title('Acúracia em função da quantidade mínima de amostras por folha \n Sem as notas')
plt.show()
```

Figura 42: Acurácias em função do parâmetro min_samples_leaf



Em seguida, de forma a se avaliar a acurácia dos dois modelos, executa-se o modelo com cada um dos conjuntos de atributos, utilizando os hiperparâmetros ótimos encontrados. O código para se fazer isso bem como o resultado estão demonstrados abaixo.

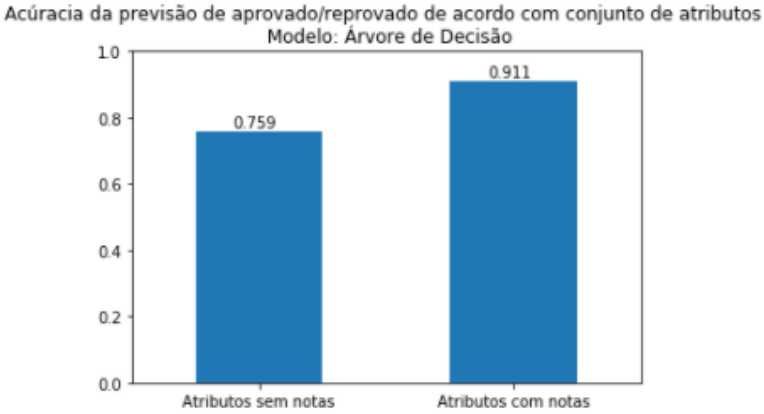
Figura 43: Código para se obter a acurácia para cada conjunto de atributos utilizando os parâmetros ótimos

```
# Sintetizando a diferença entre os dois conjuntos de atributos
    # Atributos sem as notas
treino_X, val_X, treino_notas, val_notas = train_test_split(X, alvo['Situacao'], test_size = 0.2, random_state = randint(0,5))
# Criando o modelo
notasArvoreDecisao = DecisionTreeRegressor(max_leaf_nodes=5,min_samples_leaf = 50,random_state=randint(0,5))
# Treinando o modelo
notasArvoreDecisao.fit(treino_X, treino_notas)
# Testando o modelo
previsoesSemNotas = notasArvoreDecisao.predict(val_X)
# Obtendo a acurácia do modelo
acuraciaSemNotas = accuracy_score(val_notas, previsoesSemNotas.round())

    # Atributos com as notas
treino_X, val_X, treino_notas, val_notas = train_test_split(XcomNotas, alvo['Situacao'], test_size = 0.2, random_state = 1)
# Criando o modelo
notasArvoreDecisao1 = DecisionTreeRegressor(max_leaf_nodes=9,min_samples_leaf = 20,random_state=1)
# Treinando o modelo
notasArvoreDecisao1.fit(treino_X, treino_notas)
# Testando o modelo
previsoesComNotas = notasArvoreDecisao1.predict(val_X)
# Obtendo a acurácia do modelo
acuraciaComNotas = accuracy_score(val_notas, previsoesComNotas.round())

fig, axs = plt.subplots()
# Plotando os resultados de acuracia
pd.Series(data = [acuraciaSemNotas,acuraciaComNotas],
          index = ['Atributos sem notas','Atributos com notas']).plot(kind = 'bar', ax = axs)
axs.set_title('Acúracia da previsão de aprovado/reprovado de acordo com conjunto de atributos \n Modelo: Árvore de Decisão')
axs.set_ylim(0,1)
axs.text(-0.1,acuraciaSemNotas +0.01, str(round(acuraciaSemNotas,3)))
axs.text(0.9, acuraciaComNotas + 0.01, str(round(acuraciaComNotas,3)))
axs.set_xticklabels(labels = ['Atributos sem notas','Atributos com notas'], rotation = 0)
plt.show()
```

Figura 44: Acurácia obtida para cada conjunto de atributos



Como se pode ver, como era de se esperar, o modelo com as notas apresenta uma acurácia maior. Contudo, o ganho de acurácia do modelo sem as notas em relação a um chute cego (67% dos alunos são aprovados em matemática) é também considerável.

Em seguida, como demonstram as Figuras 45 e 46) para melhor se compreender o desempenho dos modelos, plotou-se suas matrizes de confusão, bem como um relatório com as principais métricas de acurácia.

Figura 45: Obtendo as matrizes de confusão e relatório de acurácia

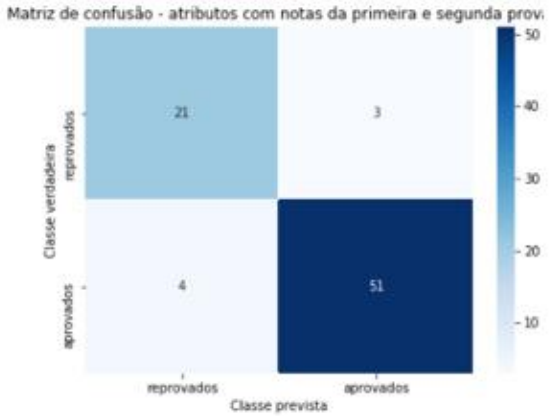
```
fig, axs = plt.subplots(1, 2, figsize=(15, 5))
# Plotando matriz de confusão dos atributos sem as notas
cmSemNotas = confusion_matrix(val_notas, previsoesSemNotas.round())
sns.heatmap(cmSemNotas, annot=True, fmt='g', ax=axs[0], cmap='Blues')
axs[0].set_xlabel('Classe prevista');axs[0].set_ylabel('Classe verdadeira')
axs[0].set_title('Matriz de confusão - atributos sem notas da primeira e segunda prova')
axs[0].xaxis.set_ticklabels(['reprovados', 'aprovados'])
axs[0].yaxis.set_ticklabels(['reprovados', 'aprovados'])

# Plotando matriz de confusão dos atributos com as notas
cmComNotas = confusion_matrix(val_notas, previsoesComNotas.round())
sns.heatmap(cmComNotas, annot=True, fmt='g', ax=axs[1], cmap='Blues')
axs[1].set_xlabel('Classe prevista');axs[1].set_ylabel('Classe verdadeira')
axs[1].set_title('Matriz de confusão - atributos com notas da primeira e segunda prova')
axs[1].xaxis.set_ticklabels(['reprovados', 'aprovados'])
axs[1].yaxis.set_ticklabels(['reprovados', 'aprovados'])

plt.show()

# Imprimindo relatório dos resultados
print('Atributos sem notas da primeira e segunda provas \n',
      classification_report(val_notas, previsoesSemNotas.round()))
print('Atributos com notas da primeira e segunda provas \n',
      classification_report(val_notas, previsoesComNotas.round()))
```

Figura 46: Matrizes de confusão e relatório de acurácia para cada conjunto de atributos



Atributos sem notas da primeira e segunda provas

	precision	recall	f1-score	support
0	0.68	0.54	0.60	24
1	0.82	0.89	0.85	55
accuracy			0.78	79
macro avg	0.75	0.72	0.73	79
weighted avg	0.78	0.78	0.78	79

Atributos com notas da primeira e segunda provas

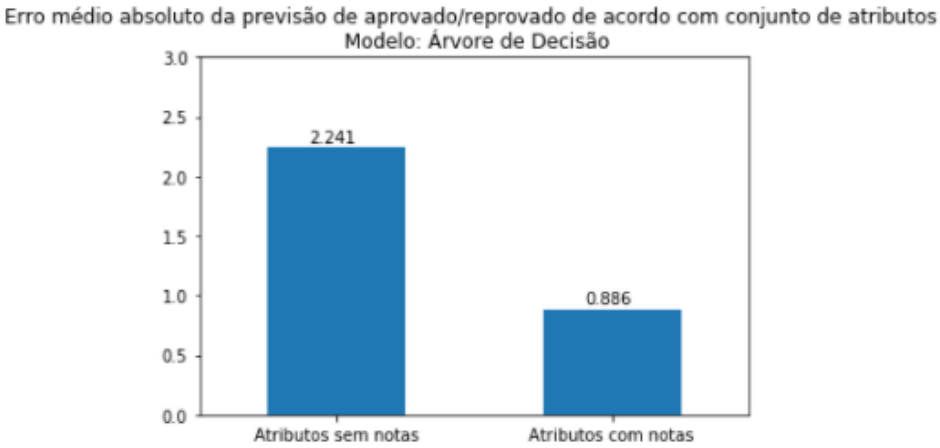
	precision	recall	f1-score	support
0	0.84	0.88	0.86	24
1	0.94	0.93	0.94	55
accuracy			0.91	79
macro avg	0.89	0.90	0.90	79
weighted avg	0.91	0.91	0.91	79

O que a matriz de confusão e o recall demonstram é que o modelo sem as notas possui grande dificuladade em detectar os alunos reprovados, o que faz com que sua acurácia global diminua.

6.3. Prevendo o valor das notas

Para se prever a nota em si do aluno, adotou-se o mesmo procedimento de otimização dos hiperparâmetros. A figura a seguir demonstra o erro médio absoluto da previsão feita com base em cada conjunto de atributos – com ou sem as notas.

Figura 47: Erro médio absoluto na estimativa da nota do aluno para cada conjunto de atributos



Embora o erro absoluto médio é relativamente baixo para o modelo que utilizou os atributos sem as notas, isso deve ser encarado com desconfiança, pois, como se viu anteriormente esse modelo prevê bem os alunos que vão passar, e mal

os que serão reprovados, então, desconfia-se que há uma alta precisão na estimativa das notas dos alunos que passaram (que são maioria e acabam por influenciar o erro) e uma precisão menor para os que tiraram notas baixas.

7. Considerações finais e sugestões de trabalhos futuros

O modelo de árvore de decisão conseguiu prever com uma alta previsão os alunos que passariam ou seria reprovados quando recebeu como entrada a nota do aluno em uma das provas anteriores à final. Esse é um resultado interessante, mas sabe-se que seria interessante fazer uma boa previsão sem qualquer informação de desempenho para possivelmente fazer um melhor acompanhamento com os alunos com potencial de serem reprovados desde o início do ano. Para tal, deve-se procurar obter outros atributos mais discriminatórios do desempenho do aluno. O que se viu na etapa de exploração de dados, especialmente nas Figuras 21 a 23, e também pelas informações mútuas levantadas é de forma geral os atributos não são muito discriminatórios do desempenho do aluno. Obteve-se algum êxito em criar atributos com maior potencial de influência no desempenho utilizando técnicas de desempenho, mas ainda assim se conseguiu apenas uma pequena melhora em relação à capacidade de previsão do modelo sem utilização das notas do que se fosse feito um chute cego se o aluno passou ou não.

Como sugestões de trabalho futuro, seria interessante utilizar mais técnicas de engenharia de atributos nos dados, tal como a *Principal Component Analysis*. Também pode ser promissor se utilizar outros modelos de *Machine Learning* mais sofisticados do que o modelo de árvore de decisão.

De qualquer forma, crê-se que pelo que foi apresentado no trabalho, cumpriu-se com objetivo de enriquecer a discussão e o entendimento da aplicação de técnicas de *Business Inteligence* à previsão do desempenho acadêmico, tarefa muito socialmente relevante.

8. Links

Link para o vídeo com o briefing do trabalho:
https://youtu.be/McOdiDkKK7E

Link para o repositório contendo todos *Jupyter Notebooks* mencionados no trabalho:
https://github.com/lucasrochex/TCC_Estimativa-Desempenho-Academico

Link para página onde os dados foram obtidos:
https://www.kaggle.com/uciml/student-alcohol-consumption

REFERÊNCIAS

CORTEZ. Paulo, SILVA. Alice, **Using Data Mining to Predict Secondary School Perfomance.** Guimares, 2008.

.