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

Guilherme Siqueira Simões

ANÁLISE DO PERFIL DOS CANDIDATOS AO FIES VIA ALGORITMOS DE AGRUPAMENTO

Belo Horizonte
2023

Guilherme Siqueira Simões

ANÁLISE DO PERFIL DOS CANDIDATOS AO FIES VIA ALGORITMOS DE AGRUPAMENTO

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
2023

SUMÁRIO

1. Introdução

1.1. Contextualização

A Constituição Brasileira de 1988 determina que a educação é um direito de todos e dever do Estado e da família. Seu artigo 208, parágrafo V, afirma que é dever do Estado garantir o acesso aos níveis mais elevados de ensino. Alinhado a este dever foi instituído o programa do Ministério da Educação (MEC) chamado FIES (Fundo de Financiamento Estudantil) pela Lei nº 10.260, de 12 de julho de 2001 (antecedido pela Medida Provisória nº 1.865-4, de 26 de agosto de 1999).

O FIES tem como objetivo financiar a graduação de estudantes em cursos de instituições de ensino superior particulares com avaliação positiva pelo MEC avaliados pelo Sistema Nacional de Avaliação da Educação Superior (Sinaes) e ofertados por instituições de educação superior não gratuitas aderentes ao programa. Ele oferece empréstimos a juros baixos para estudantes que não têm condições financeiras de arcar com os custos do curso. O financiamento é concedido mediante a assinatura de um contrato entre o estudante e a Caixa Econômica Federal, que é a agente financeira do programa.

1.2. Regras do FIES

Para participar do FIES, o estudante precisa atender cumulativamente, às seguintes condições:

- Ter participado do Exame Nacional do Ensino Médio (ENEM) a partir da edição de 2010 e obtido média aritmética das notas nas cinco provas igual ou superior a 450 pontos, e nota na prova de redação superior a zero.
- Possuir renda familiar mensal bruta per capita de até 3 salários-mínimos.

Na sua inscrição ao processo seletivo do FIES, o candidato deve obrigatoriamente informar:

- Seu número de registro no Cadastro de Pessoa Física – CPF.
- Correio eletrônico (e-mail) pessoal válido.

- Nomes dos membros do seu grupo familiar, o número de registro no CPF dos membros do seu grupo familiar com idade igual ou superior a 14 (quatorze) anos, as respectivas datas de nascimento e, se for o caso, a renda bruta mensal de cada componente do grupo familiar.
- Os parâmetros que definem o grupo de preferência.
- A ordem de prioridade das 3 (três) opções de curso/turno/local de oferta entre as disponíveis no referido grupo.

O edital com as regras de inscrição para o FIES na oferta 2020/2 está disponível em http://portalfies.mec.gov.br/arquivos/edital_43_17062020.pdf.

1.3. O problema proposto

Pelos próprios requisitos do FIES, há um agrupamento de estudantes pelo critério de renda e nota no ENEM. O objetivo deste trabalho é aplicar técnicas de agrupamento para entender dentre este universo de candidatos ao FIES, quais são suas diferenças/similaridades e assim ajudar a elaborar melhorias ao programa do MEC.

Para descrever o problema que se propõe a resolver através da Ciência de Dados e uma visão de solução, será utilizada a técnica dos 5-Ws, que consiste em responder às seguintes questões.

Why? (Por que esse problema é importante?): O programa FIES é um importante instrumento de inclusão social, tendo em vista que proporciona acesso ao ensino superior a pessoas de baixa renda, sendo que o nível de instrução de ensino é um forte fator de promoção de nível social. Ao longo de mais de 20 anos de programa é natural que seus critérios sejam revistos de tempos em tempos para análise de eventuais ajustes no intuito de melhorar sua eficácia. O entendimento dos diferentes grupos do universo de candidatos é uma informação que ajuda neste sentido.

Who? (De quem são os dados analisados?): Os dados são do governo federal do Brasil, publicados pelo MEC e relativos às inscrições de candidatos ao processo seletivo do FIES.

What? (Quais os objetivos com essa análise? O que será analisado?): Os dados de inscrição dos candidatos ao FIES serão analisados com técnicas de agrupamento com o objetivo de identificar os diferentes grupos de candidatos.

Where? (Quais os aspectos geográficos da análise): Os dados são de abrangência nacional pois envolvem candidatos de todos os estados do Brasil.

When? (Qual o período está sendo analisado?): Os dados correspondem à oferta 2020/2, que é o segundo semestre de 2020. O salário-mínimo vigente à época era R$1.045,00.

1.4. Objetivos

Agrupamento, clusterização, ou análise de cluster, é uma técnica de aprendizado não supervisionado em ciência de dados (o que significa que não é necessário etiquetar os dados) e que envolve a identificação de grupos (ou clusters) de objetos ou observações semelhantes dentro de um conjunto de dados. A ideia básica por trás da técnica é agrupar dados em subconjuntos homogêneos, onde os objetos dentro de um cluster são mais semelhantes entre si do que com os objetos em outros clusters. É útil para identificar padrões em dados, para classificar dados e para entender as relações entre dados.

Existem vários algoritmos de agrupamento, cada um com suas vantagens e desvantagens. Alguns dos algoritmos mais populares incluem o k-means, o DBSCAN e o hierárquico.

A clusterização é amplamente utilizada em diversas áreas, como marketing, genômica, análise de imagens, detecção de fraudes, entre outras. Alguns exemplos de aplicações incluem a segmentação de clientes para campanhas de marketing direcionadas, a identificação de genes relacionados em um conjunto de dados genômicos e a identificação de padrões em imagens para reconhecimento de objetos.

O objetivo deste trabalho é analisar a identificação de perfis dentre os candidatos ao FIES, considerando sua primeira opção de curso. Apesar do critério do programa já estabelecer um recorte dos candidatos pelo nível de renda familiar, deseja-se identificar

quais são os diferentes grupos de candidatos que se candidatam ao programa. Cabe ressaltar que o FIES possui um comitê gestor, que tem como uma das atribuições supervisionar a execução das operações e, que através de sua resolução 26 estabeleceu 11 indicadores de desempenho que deverão ser objeto de acompanhamento, mas nenhum deles aborda o perfil do candidato ao programa.

2. Coleta de Dados

Neste trabalho foram usadas fontes de dados distintas, uma contendo dados relativos às inscrições de candidatos ao FIES e outras duas contendo informação dos municípios brasileiros com o objetivo de enriquecer a primeira fonte de dados com dados de latitude e longitude dos municípios.

A primeira fonte de dados foi obtida em 04/03/2023 no portal de dados abertos mantidos pelo MEC. São dados de inscrições no FIES da oferta 2020/2 disponíveis em https://dadosabertos.mec.gov.br/images/conteudo/fies/2020/relatorio_inscricao_dados_abertos_fies_22020.csv em formato CSV contendo 212.404 linhas com a seguinte estrutura.

Nome da coluna	Descrição	Tipo
Ano do processo seletivo	Ano da oferta do FIES	Numérico, 4 dígitos (AAAA)
Semestre do processo seletivo	Semestre do ano da oferta do FIES	Numérico (1 ou 2)
ID do estudante	Identificação do candidato ao FIES	Inteiro, 9 dígitos
Sexo	Gênero do candidato	M – Masculino F – Feminino
Data de nascimento	Data de nascimento	Data (DD/MM/AA)
UF de residência	Estado onde mora o candidato	Sigla do estado, 2 letras
Município de residência	Cidade onde mora o candidato	Texto
Etnia/Cor	Cor declarada pelo	Amarela

	candidato	Branca Indígena Parda Preta
Pessoa com deficiência?	Se o candidato possui alguma deficiência	Sim Não
Tipo de escola no ensino médio	Se o candidato frequentou escola pública no ensino médio	Sim Não Parcial
Ano conclusão ensino médio	Ano conclusão ensino médio	Numérico, 4 dígitos (AAAA)
Concluiu curso superior?	Se o candidato já concluiu algum curso superior	Sim Não
Professor rede pública ensino?	Se o candidato é professor da rede pública	Sim Não
Nº de membros Grupo Familiar	Tamanho da família do candidato (além do próprio)	Inteiro
Renda familiar mensal bruta	Soma da renda de todos os membros da família do candidato	R$
Renda mensal bruta per capita	Renda familiar dividida pela quantidade de membros da família do candidato	R$
Região grupo de preferência	Região do Brasil na qual o candidato deseja concorrer à uma vaga	Centro-Oeste Norte Nordeste Sudeste Sul
UF	Estado do Brasil na qual o candidato deseja concorrer à uma vaga	Sigla do estado, 2 letras
Cod. Microrregião	Código da microrregião do	Numérico, 5 dígitos

	Brasil na qual o candidato deseja concorrer à uma vaga	
Microrregião	Descrição da microrregião do Brasil na qual o candidato deseja concorrer à uma vaga	Texto
Cod. Mesorregião	Código da mesorregião do Brasil na qual o candidato deseja concorrer à uma vaga	Numérico, 4 dígitos
Mesorregião	Descrição da mesorregião do Brasil na qual o candidato deseja concorrer à uma vaga	Texto
Conceito de curso do GP	Nota da IES no Sistema Nacional de Avaliação da Educação Superior (SINAES) do MEC	Numérico, 1 dígito (3 a 5) Nulo (para cursos que ainda não possuam avaliação no SINAES)
Área do conhecimento	Descrição da área do conhecimento da especialidade do curso	Texto
Subárea do conhecimento	Descrição da subárea do conhecimento da especialidade do curso	Texto
Cod. do Grupo de preferência	Opção de grupo de preferência escolhido pelo candidato	Numérico, 6 dígitos
Nota Corte Grupo Preferência	Nota de corte do ENEM estabelecida para o grupo de preferência	999,99
Opções de cursos da inscrição	Ordem de prioridade indicada pelo candidato para o curso escolhido	Numérico: 1, 2 ou 3
Nome mantenedora	Razão Social da IES que	Texto

	oferta o curso de interesse do candidato	
Natureza Jurídica Mantenedora	Tipo de natureza jurídica da mantenedora	Texto
CNPJ da mantenedora	Cadastro Nacional de Pessoa Jurídica da mantenedora	Numérico
Nome da IES	Nome da Instituição de Ensino Superior (IES) ofertante do curso de interesse do candidato	Texto
Código e-MEC da IES	Identificação de registro da IES junto ao MEC	Inteiro
Organização Acadêmica da IES	Corresponde ao nível de autonomia da IES perante o credenciamento junto ao MEC	Faculdade Centro Universitário Universidade
Município da IES	Município sede da Instituição de Ensino Superior ofertante do curso de interesse do candidato	Texto
UF da IES	Estado sede da Instituição de Ensino Superior ofertante do curso de interesse do candidato	Texto, 2 letras
Nome do Local de oferta	Nome do campus da IES onde o curso é ofertado	Texto
Código do Local de Oferta	Código do campus da IES onde o curso é ofertado	Numérico
Munícipio do Local de Oferta	Município do local onde o curso de interesse do candidato é ofertado	Texto

UF do Local de Oferta	Estado do local onde o curso de interesse do candidato é ofertado	Texto, 2 letras
Código do curso	Código do curso de interesse do candidato	Numérico
Nome do curso	Nome do curso de interesse do candidato	Texto
Turno	Turno do dia em que as aulas do curso são ofertadas	Integral Matutino Vespertino Noturno
Grau	Grau do curso	Trimestral Semestral Anual
Conceito	Conceito do curso na avaliação do MEC	Numérico, 1 dígito
Média nota Enem	Média da nota do candidato no ENEM	999,99
Ano do Enem	Ano em que o candidato prestou o ENEM usado como nota na inscrição	Numérico, 4 dígitos (AAAA)
Redação	Nota do ENEM obtida pelo candidato na redação	Numérico, 4 dígitos
Matemática e suas Tecnologias	Nota do ENEM obtida pelo candidato em Matemática	9999,99
Linguagens, Códigos e suas Tec	Nota do ENEM obtida pelo candidato em Linguagens	9999,99
Ciências Natureza e suas Tec	Nota do ENEM obtida pelo candidato em Ciências da Natureza	9999,99
Ciências Humanas e suas Tec	Nota do ENEM obtida pelo	9999,99

	candidato em Ciências da Natureza	
Situação Inscrição Fies	Situação final da inscrição do candidato no FIES	Texto
Percentual de financiamento	Percentual de financiamento do curso obtido pelo candidato que possui a situação de inscrição “contratada”	Porcentagem
Semestre do financiamento	Semestre do curso a partir do qual o financiamento foi concedido	Numérico
Qtde semestre financiado	Quantidade de semestres financiados pelo candidato	Numérico

A descrição de alguns dos atributos da tabela anterior foi baseada no dicionário de dados fornecido pelo MEC em http://dadosabertos.mec.gov.br/fies/item/81-dicionario-de-dados. Dados de inscrições em outras ofertas podem ser obtidos em https://dados.gov.br/dados/conjuntos-dados/mec-fundo-de-financiamento-estudantil-fies.

A segunda fonte de dados foi obtida em 04/03/2023 de https://github.com/kelvins/Municipios-Brasileiros em formato CSV contendo 5.570 linhas com a seguinte estrutura.

Nome da coluna	Descrição	Tipo
codigo_ibge	Código do município no IBGE	Numérico, 7 dígitos
nome	Nome do município	Texto
latitude	Latitude do município	Numérico
longitude	Longitude do município	Numérico
capital	Indicador se é capital	0 – não é capital 1 – é capital

codigo_uf	Código do estado do município no IBGE	Numérico
siafi_id	Código do município no SIAFI (Sistema Integrado de Administração Financeira do Governo Federal)	Numérico
ddd	Código telefônico de área do município	Numérico, 2 dígitos
fuso_horario	Fuso horário do município	Texto

Há um problema para o enriquecimento dos dados apenas com estas duas fontes de dados: municípios com mesmo nome em estados diferentes. Nos dados fornecidos pelo MEC o município é identificado de forma única pelo seu nome e a sigla do estado. Na segunda fonte de dados que contempla latitude e longitude dos municípios o município pode ser identificado de forma única de três maneiras: pelo código IBGE, pelo código SIAFI ou pelo seu nome e código IBGE (não a sigla) da sua UF.

Portanto, é necessário buscar uma terceira fonte de dados para se conseguir estabelecer de forma unívoca um município na base do FIES com sua respectiva latitude e longitude na segunda base. Optou-se neste caso por usar a base de dados da STN (Secretaria do Tesouro Nacional) que contempla o código SIAFI, código IBGE, nome do município e sigla da UF. Os dados foram obtidos em 04/03/2023 de https://www.tesourotransparente.gov.br/ckan/dataset/lista-de-municipios-do-siafi/resource/eebb3bc6-9eea-4496-8bcf-304f33155282 em formato CSV contendo 5.589 linhas com a seguinte estrutura.

Nome da coluna	Descrição	Tipo
id_siafi	Código do município no SIAFI	Numérico
coluna 2	desconhecida	Numérico
nome_mun	Nome do município	Texto
sigla_uf	Sigla da UF	Texto
cod_ibge	Código do município no IBGE	Numérico

Infelizmente estes dados são fornecidos sem o nome das colunas na primeira linha e não há um dicionário de dados para o arquivo no portal de dados abertos da STN (vide https://www.tesourotransparente.gov.br/ckan/dataset/lista-de-municipios-do-siafi). Porém em uma inspeção visual e comparando com o conteúdo do município de mesmo nome na segunda base de dados foi possível identificar o significado das colunas relevantes para a junção do município nos dados do FIES com dados de latitude e longitude.

3. Processamento/Tratamento de Dados

Os scripts para a realização deste trabalho foram escritos em Python, na plataforma Jupiter Notebook presente na distribuição Anaconda (https://www.anaconda.com/). Os dados foram carregados para objetos do tipo DataFrame da biblioteca Pandas para o pré-processamento e depois convertidos para objetos do tipo Array da biblioteca Numpy para o processamento pelos algoritmos de agrupamento.

3.1 Carga de Dados

O arquivo de inscrições do FIES foi carregado para um dataframe via read_csv(). Como o arquivo é delimitado pelo separador ponto e vírgula, fez-se uso do parâmetro sep = ';'. A codificação dos caracteres é Windows-1252, provavelmente pelo uso de acentos em alguns dados, por isso fez-se uso do parâmetro encoding='cp1252'. E por fim, os valores fracionários estão em formato brasileiro, com a separação da parte fracionária pela vírgula em vez do ponto (do formato americano), por isso usaram-se os parâmetros decimal=',' e thousands='.' para que o tipo das colunas fosse carregado já como float64 em vez object. Foram carregadas 212.404 linhas.

```
df_fies = pd.read_csv(pasta + arq_fies, sep = ';', encoding='cp1252', decimal=',', thousands='.')
```

O arquivo de municípios com latitude e longitude foi carregado para outro dataframe também via read_csv(). Como o arquivo é delimitado pelo separador vírgula, fez-se uso do parâmetro sep = ','. Já na própria carga foram desprezadas duas colunas: a que indica que o

município é uma capital e a que contém o código IBGE da sua UF. Foram carregadas 5.570 linhas.

```
df_latlong_mun = pd.read_csv(pasta + arq_latlong_mun, sep = ',', usecols=[0,1,2,3,6])
```

O arquivo de municípios com código SIAFI e IBGE foi carregado para outro dataframe também via `read_csv()`. Como o arquivo é delimitado pelo separador ponto e vírgula, fez-se uso do parâmetro `sep = ';'`. O arquivo foi disponibilizado sem cabeçalho, a primeira linha sendo tratada já como dado. Por isso usou-se o parâmetro `header=None`. E usou-se o parâmetro `names` para nomear as colunas. Já na própria carga foi desprezada a segunda coluna, cujo significado não possível descobrir, mas tampouco é relevante para o propósito deste trabalho. Foram carregadas 5.589 linhas.

```
df_stn_mun = pd.read_csv(pasta + arq_stn_mun, sep = ';', header=None, usecols=[0,2,3,4], names=['siafi_id', 'nome_mun', 'sigla_uf', 'codigo_ibge'])
```

Embora as duas bases de municípios não contenham a mesma quantidade de registros, isso não é um problema desde que se consiga enriquecer com latitude e longitude todos os municípios presentes na base de inscrições do FIES.

3.2 Remoção de dados duplicados

Para os três dataframes carregados executou-se o método `drop_duplicates(keep='first', inplace=True)` com o objetivo de eliminar registros duplicados. Não houve duplicidade nos dados de município. E curiosamente foram eliminadas 2.902 linhas nos dados do FIES.

Pelas regras do FIES o candidato pode selecionar até três opções de curso, em ordem de prioridade, quando isso ocorre há a repetição do registro do candidato para cada opção de curso selecionada. Como o objetivo é analisar os dados de inscrição da opção de curso mais prioritária, usou-se o método `drop(df_fies[df_fies['Opções de cursos da`

`inscrição'] != 1].index, inplace=True)` para remover todas as linhas relativas às opções de curso diferente da primeira. Foram removidos 101.610 registros.

3.3 Seleção inicial de atributos do FIES

Dos 57 atributos da base do FIES, nem todos são relevantes para o propósito deste trabalho, logo, foi feita sua eliminação do dataframe através do método `del df['coluna']`, de forma a simplificar o processamento das etapas seguintes. Segue a relação das colunas eliminadas e o respectivo motivo.

- Opções de cursos da inscrição: pois restaram apenas os registros relativos à opção 1.
- Ano e semestre do processo seletivo são constantes em toda base.
- ID do estudante não é atributo relevante para os algoritmos de agrupamento.
- Atributos relativos à situação do financiamento obtido, pois a intenção é analisar o perfil dos candidatos, independente se conseguiram ou não obter o financiamento: Situação Inscrição Fies, Percentual de financiamento, Semestre do financiamento, Qtde semestre financiado.
- Atributos da IES (Instituição de Ensino Superior) ofertante do curso, tendo como premissa de que o interesse do candidato é pelo curso não pela IES: Nome mantenedora, Natureza Jurídica Mantenedora, CNPJ da mantenedora, Código e-MEC da Mantenedora, Nome da IES, Código e-MEC da IES, Organização Acadêmica da IES, Município da IES, UF da IES, Nome do Local de oferta, Código do Local de Oferta.
- Atributos criados pelo IBGE para agrupar municípios, levando em conta fatores como proximidade geográfica, características econômicas, sociais e culturais em comum. Isto é utilizado para a formação de grupos de preferência, que priorizam estudantes que residem em regiões com baixo IDHM. Considerou-se que há redundância destes atributos com o município e UF de residência do candidato:

Região grupo de preferência, UF, Microrregião, Cod.Microrregião, Mesorregião, Cod.Mesorregião.

- Atributos relativos à área temática do curso de graduação do candidato (Área e subárea do conhecimento), cuja classificação é definida pela CAPES. Como será utilizado o curso do candidato no algoritmo de classificação, optou-se por eliminar estes atributos, pois podem introduzir um viés no algoritmo de classificação reduzindo o peso da escolha do curso pelo candidato. Por exemplo, cursos diferentes, mas de mesma área e subárea tendem a aproximar candidatos (viés da CAPES).

- O grupo de preferência é derivado do estado, município e o curso que o candidato quer estudar. Por ser um atributo derivado de outros, é redundante.

- Avaliou-se se código do curso e nome do curso são equivalentes pelo método `.nunique()` e se percebeu que a quantidade de valores únicos de código do curso é muito superior aos valores únicos de nome do curso. Em uma amostra de registros cujo nome do curso é DIREITO, encontrou-se 983 códigos diferentes. Para favorecer a aproximação pelo algoritmo de candidatos com curso de mesmo nome, mesmo com códigos diferentes, optou-se por manter o nome e eliminar o código do curso.

- Para a coluna Grau, analisou-se seus valores únicos para tentar entender seu significado, pelo método `.value_counts()`. Foram encontrados apenas três valores distintos: trimestral, semestral e anual. Observou-se que 98% das ocorrências são para o valor semestral, o que indica baixa variância dos valores da coluna. Tomando como premissa que esse dado não seria um fator relevante para a escolha do curso pelo candidato; como seria por exemplo a duração do curso ou valor da mensalidade (atributos inexistentes na base), optou-se por eliminar o atributo.

- Para a coluna ‘Média nota Enem’ verificou-se que o seu valor coincide com a média aritmética das 5 notas individuais do ENEM (redação, matemática, linguagens, ciências da natureza e ciências humanas. A verificação foi feita

aplicando um filtro ao dataframe no qual foi feita a diferença entre a nota do ENEM e a média aritmética das cinco notas. O resultado não foi zero para todas as ocorrências em função de questões de arredondamento no cálculo da nota do ENEM, mas em todos os casos a diferença foi inferior a 1. Logo a nota do ENEM foi eliminada por ser dado redundante, derivado das outras cinco notas. Tomou-se também o cuidado de verificar que não havia nenhuma ocorrência cuja nota média de ENEM fosse inferior a 450, que é a nota mínima exigida pelo FIES, e que poderia indicar algum registro com erro nos dados.

Há duas colunas para conceito de curso: a do curso selecionado (‘Conceito’) e a do grupo de preferência ('Conceito de curso do GP’). O conceito de curso é a nota de qualidade dada pelo MEC aos cursos de graduação de IES credenciadas. Podem ser financiados os cursos de graduação com conceito maior ou igual a 3, numa escala de 1 a 5. Os cursos que ainda não possuam avaliação, mas que estejam autorizados para funcionamento, podem participar do FIES.

Como essas colunas foram carregadas como tipo object, verificou-se os valores distintos existentes pelo método `.value_counts()`. Além dos valores 3, 4 e 5, há também o valor 'Autorizado'. Decidiu-se substituir este valor por zero e transformar a coluna em numérica. Em seguida analisou-se relação entre os dois atributos via gráfico de dispersão.

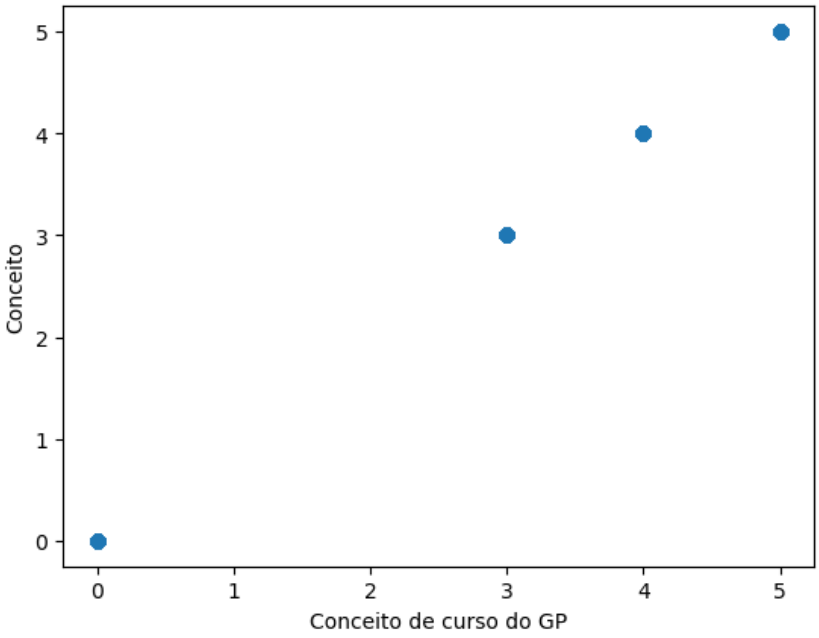


Gráfico de dispersão entre conceito do curso e conceito do curso do GP: a relação entre os valores de ambas as colunas mostra coincidência em todas as ocorrências.

Como ambas as colunas possuem valores idênticos, eliminou-se uma delas, a do conceito de curso do GP.

3.4 Transformação de dados

Observando nos primeiros registros o formato carregado para a data de nascimento do candidato com o método `.head`, constata-se que o ano possui apenas dois dígitos. Porém há datas em séculos diferentes. Ao converter a coluna de object para datetime com o método pd.to_datetime, observou-se o bug do milênio ao listar as datas em ordem decrescente com o método `.sort_values('Data de Nascimento', ascending=False)`, encontrando datas de nascimento superiores à data atual. Para corrigir o bug do milênio na conversão de data do ano com 2 dígitos, foram subtraídos 100 anos das datas de nascimento superiores ao ano da oferta do FIES (2020).

```
df_fies.loc[df_fies['Data de Nascimento'] > cutoff_date, 'Data de Nascimento'] -= pd.DateOffset(years=100)
```

Em seguida buscou-se registros em que a data de nascimento era maior ou igual ao ano de conclusão do ensino médio. Foi encontrada uma ocorrência, e a opção foi pela eliminação do registro. Também se pesquisou registros cuja ano de nascimento era maior ou igual ao ano de conclusão do ensino médio, mas não foi encontrada nenhuma ocorrência.

Em geral os algoritmos de machine learning embutem diversas operações matemáticas, e transformar atributos para numéricos acaba sendo necessário. Neste caso será usada a idade do candidato em vez de sua data de nascimento. A idade foi calculada em anos considerando quando a inscrição ao FIES 2020/2 foi aberta, em 09/08/2020.

4. Análise Exploratória

Após a eliminação de atributos desnecessários ao processamento e transformação da data de nascimento em idade, foi executado o método `.describe()` para obter uma visão descritiva das estatísticas das colunas dos dataframes em busca de algum dado com comportamento estranho.

Para os dois dataframes de município, nada de estranho foi encontrado. Não há valores nulos nas colunas. Os dados estatísticos das colunas numéricas do SIAFI e IBGE não são relevantes pois os dados são categóricos nominais. A duplicidade de nomes de municípios já era esperada, tendo em vista que há municípios de mesmo nome em UF distintas. E os valores de latitude e longitude estão dentro da faixa: [-90 a 90] e [-180 a 180] respectivamente.

Para o dataframe do FIES, a quantidade de linhas após o processamento das etapas anteriores ficou em 107.891. Observando as estatísticas das colunas numéricas, para a maioria se observou uma faixa de valores normal. Para a idade do candidato [15 a 74], sem valores absurdos (ex.: <=0 ou >100). Para o ano de conclusão do ensino médio [1969 a 2020], sem valores absurdos (>2020, superior ao ano da oferta do FIES). Verificou-se que o menor ano (1969) é também do candidato com maior idade (74 anos). Para o número de membros do grupo familiar [0 a 13], sem valores absurdos (ex.: <=0 ou >20). Para as notas do ENEM, sem valores absurdos (ex.: <=450 ou >1.000). Para o ano do ENEM do candidato [2010 a 2020], sem valores absurdos (<2010 ou >2020, superior ao ano da oferta do FIES). Não há regra entre o ano do ENEM que o candidato prestou o exame e o ano de conclusão do ensino médio. Treineiros podem fazer o ENEM antes de concluir o ensino médio. O conceito do

curso [3 a 5], dentro das regras do FIES, sendo que o valor 0 corresponde aos cursos autorizados que ainda não possuem avaliação.

Observou-se que na coluna de renda familiar há valores nulos e que o valor mínimo do número de membros do grupo familiar é zero. Foi executado um filtro nas linhas cujo número de membros é zero e se constatou que em todos os casos a renda familiar é nula, porém há valores na renda per capita. Isto indica que o MEC não considera o próprio candidato como parte do número de membros do grupo familiar.

Foi feita também a verificação em sentido inverso, de que para todas as linhas cuja renda familiar é nula, o número de membros do grupo familiar é zero. Isso indica que a renda familiar mensal é informada apenas quando há outros membros no grupo familiar. Observa-se também que a renda per capita média não é zero, o que indica que a renda per capita computa a renda do candidato e a renda familiar não.

Verificou-se também qual o número de membros do grupo familiar quando a renda familiar é zero. Foram encontradas 21 ocorrências, e em todas o número de membros familiares é maior que zero e renda per capita é zero. Isso indica que o candidato informou membros da família, maior de 14 anos e sem renda e que ele próprio não possui renda.

Testou-se em quantas linhas há coincidência entre a renda familiar e a renda per capita multiplicada pelo número de membros. A coincidência existe apenas quando ambas as colunas estão zeradas, o que indica que o candidato não compõe o número de membros.

Decidiu-se então preencher a coluna renda familiar com zeros nos casos onde seu valor seja nulo (neste caso o número de membros do grupo já é zero), usando o método `.fillna(0, inplace=True)`.

Outra verificação foi de quantos registros há com renda familiar menor que a renda per capita. Em todos a renda familiar é zero, o que confirma que a renda do candidato não compõe a renda familiar. Em seguida verificou-se em quantas linhas há coincidência entre a renda familiar e a renda per capita multiplicada pelo número de membros + 1 (o próprio candidato). Há diferenças de centavos ocasionado por arredondamento em algumas linhas, portanto se considerou que não há coincidência apenas quando a diferença é superior a R$1.

O resultado foi de 93.725 linhas. Para os 14.166 registros onde não houve essa coincidência, a renda familiar e o número de membro era zero. Se for considerado no tamanho da família somente o próprio candidato, a renda familiar seria equivalente à renda per capita. Logo a renda familiar é uma informação redundante, derivada do número de membros e a renda per capita e, portanto, a coluna foi desprezada.

Na sequência analisou-se as colunas não numéricas do dataframe, usando o método `.describe(include=[object])`. Quatro destas colunas contemplam valores binários. Usou-se o método `.value_counts()` para se investigar os valores distintos existentes. A única questão anômala que se observou foi no nome do município a acentuação e no caso de nomes com apóstrofo, este aparece com o caractere de interrogação invertido. Provavelmente houve problema na exportação dos dados pelo MEC. Decidiu-se então aplicar o método `.apply(lambda x: unidecode(x))` para eliminar acentos e caracteres especiais, já em preparação para fazer a mesclagem com dados de latitude e longitude do município pelo nome. Após esta transformação, observou-se que o caractere de interrogação invertido foi substituído pelo caractere de interrogação normal, não o apóstrofo. Aplicou-se o método `.str.replace('\?', '\'')` para fazer a substituição correta do caractere.

Outra questão interessante de notar é com relação ao município de residência do candidato e do local de oferta do curso. Os candidatos residem em 4.382 municípios distintos e buscam cursos ofertados em 514 municípios distintos. O que indica uma concentração na oferta de cursos em poucos municípios. E que para mais da metade dos candidatos (57.310) a busca é por um curso em município diferente do de sua residência. Considerando que o programa é direcionado a um público de baixa renda, percebe-se mais uma barreira econômica para o acesso à graduação, além do valor da mensalidade, há o custo de transporte (ou até mudança) para outro município.

Com base nos dados processados pode-se traçar o perfil médio do candidato ao FIES:

- É do sexo feminino, de cor parda e não possui deficiência física. O que coincide com o perfil dominante da população brasileira.
- Reside em São Paulo-SP. Normal, considerando ser a cidade e o estado mais populoso do país.
- Tem 23 anos de idade, cursou o ensino médio em escola pública, concluiu o ensino médio em 2014 e nunca concluiu o curso superior. Ou seja, o candidato termina o ensino médio na idade correta e seis anos passados ainda busca acesso ao

ensino superior. O que indica a barreira de acesso existente a este nível de educação. Ou talvez possa ter iniciado um curso superior antes e evadiu.

- Usa a nota do ENEM de 2017, o que pode indicar que ele não tentou o ENEM imediatamente após concluir o ensino médio. Ou que demorou de 2 a 3 anos até obter uma nota considerada satisfatória por ele, de média 564. Paradoxalmente a nota de corte do seu grupo de preferência é mais alta, 616. No entanto, passados 3 anos, ainda tenta acesso ao FIES. O que pode indicar que ele desistiu de tentar acesso a uma universidade pública via ENEM (porque se essa fosse a intenção, tentaria o ENEM todos os anos).
- Vive com menos que dois familiares, talvez com um dos pais (o outro falecido ou separado) ou com um cônjuge.
- A renda mensal per capita é de R$1.083, pouco mais de um salário-mínimo. Há uma distância razoável para o limite de três salários definidos para o FIES.
- Nunca foi professor da rede pública de ensino.
- Busca um curso de conceito entre 3 e 4. Seriam os cursos de conceito 5 mais caros?
- Deseja realizar o curso na própria cidade.
- No período noturno, muito provavelmente pela necessidade de trabalhar durante o dia.
- Optando pelo curso de Medicina, o mais concorrido nos vestibulares. Cabe um destaque que não há curso de medicina no período noturno, somente integral. Em todos os demais cursos, a opção majoritária é pelo período noturno, embora o curso mais escolhido seja medicina. É o perigo de se analisar dados apenas pelas médias.

4. Enriquecimento dos Dados

A primeira etapa de enriquecimento consiste na junção dos dois dataframes de municípios em um terceiro para que se possa ter o nome do município, a sigla da UF e latitude e longitude. Para isso usou-se o comando:

```
pd.merge(df_latlong_mun, df_stn_mun, on=['siafi_id', 'codigo_ibge'], how='inner')
```

Aplicou-se uma transformação no nome do município para que os caracteres ficassem todos em maiúscula e eliminação de acentuação e caracteres especiais. Não se observou problema com o apóstrofo como no nome do município na base do FIES.

O enriquecimento de dados do município com sua latitude e longitude foi feito para duas colunas da base do FIES: a do município de residência do candidato e a do município do local de oferta do curso. A chave para fazer a junção é o nome do município e a sigla da UF, já devidamente padronizados em ambos dataframes. Antes da junção as colunas do dataframe de municípios foram renomeadas para coincidir com os respectivos nomes das colunas do dataframe do FIES.

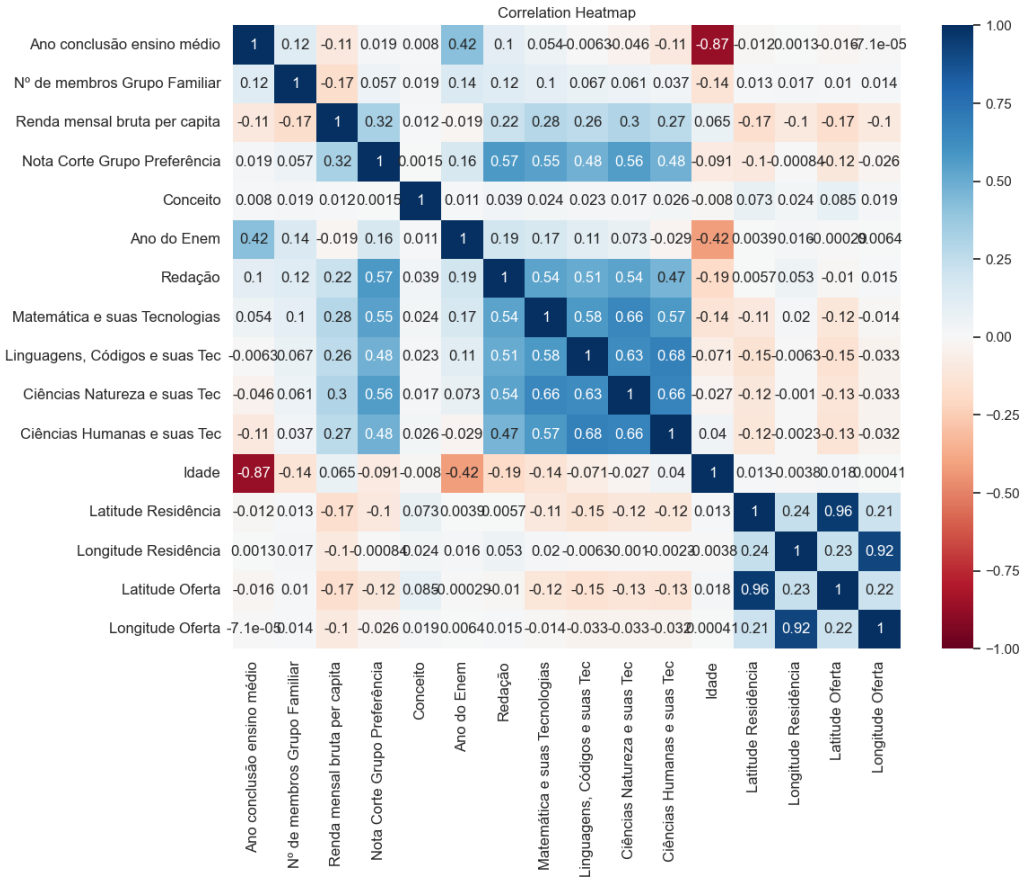
A junção foi feita pelo comando seguinte. Porém, observou-se que não foram encontradas correspondências de latitude e longitude para 26 ocorrências. Investigando estes nomes, percebeu-se problemas de grafia na base do FIES para 6 municípios. Foi feita a correção na dataframe original do FIES e executada a junção novamente, restando ainda quatro ocorrências sem correspondente latitude/longitude. Neste caso o problema estava no dataframe de municípios, que continha no nome do município o nome novo e o antigo entre parênteses para dois municípios. Foi feita a correção apenas para o nome novo e executada a junção novamente, agora com total sucesso.

```
pd.merge(df_fies, df_mun_merged, on=['Municipio de residência', 'UF de residência'], how='left')
```

A junção dos dados para o município do local de oferta do curso foi feita de forma análoga, com a ressalva de que não houve necessidade de tratamento adicional nos dados, a junção foi feita com sucesso na primeira tentativa.

Com o enriquecimento dos dados com as coordenadas geográficas do município do candidato e do município de oferta do curso selecionado, o nome do município tornou-se desnecessário. Optou-se também por eliminar os atributos relativos à UF do município, para que se privilegie o critério da distância geográfica e não o da divisão política. Por exemplo, candidatos que moram em cidades próximas deveriam ser aproximados, mesmo que as cidades sejam de UF distintas (é o caso de regiões de fronteira, por exemplo).

Após essa etapa de enriquecimento, o dataframe ficou com 24 atributos e para os atributos numéricos gerou-se o seguinte mapa de calor com a correlação entre os atributos. A correlação é calculada pelo método `.corr()` e varia entre -1 e 1. Quanto mais próxima de zero, indica baixa correlação entre as variáveis. Quanto mais próximo de 1, indica alta correlação positiva. E quanto mais próximo de -1, indica alta correlação negativa.



Mapa de calor com a correlação dos atributos numéricos do dataframe após enriquecimento.

Percebe-se uma alta correlação positiva entre latitude e longitude do município de residência do estudante e do município de oferta do curso, o que faz sentido. O candidato busca um curso na própria cidade ou em cidade próxima à sua.

Percebe-se também uma alta correlação negativa entre a idade do candidato e o ano de conclusão do ensino médio. Ou seja, quanto mais velho o candidato, mais provável que tenha terminado o ensino médio mais cedo, o que é esperado.

5. Tratamento dos atributos categóricos

A partir deste ponto do trabalho, optou-se por restringir o processamento para as uma amostra aleatória de 10.000 linhas do dataframe. Isto com a finalidade de agilizar as diversas simulações necessárias até a versão final do script Python. Segundo as regras do trabalho de conclusão de curso da PUC-Minas, a quantidade mínima de linhas para o dataset a ser usada é de 1.000. Logo, embora o resultado do processamento para todas as linhas não seja exatamente o mesmo, o raciocínio utilizado permanece igual.

Os algoritmos de agrupamento utilizados neste trabalho são baseados em cálculos de distância. Logo para o processamento dos pelos algoritmos, é necessário que todos os atributos sejam numéricos. O dataframe possui 23 atributos, sendo dos quais 7 são atributos categóricos, todos nominais. Foi feita então a codificação destes atributos via label encoder, para transformação de dados textuais em numéricos. A seguir segue exemplo usado:

```
label_encoder_Sexo = LabelEncoder()

fies_lab_encoded[:,0] = label_encoder_Sexo.fit_transform(fies_lab_encoded[:,0])
```

E na sequência foi feita a codificação de atributos via one hot encoder para os atributos categóricos não binários, como segue:

```
onehotencoder_fies = ColumnTransformer(transformers=[('OneHot', OneHotEncoder(),
[1,3,10,11])], remainder='passthrough')

fies_ohe_encoded = onehotencoder_fies.fit_transform(fies_lab_encoded)
```

6. Padronização dos dados

Como dito na seção anterior, os algoritmos de agrupamento utilizados neste trabalho são baseados em cálculos de distância. Logo para que o algoritmo não pondere de forma diferenciada um atributo em relação aos demais, é necessário que os dados sejam normalizados. Por exemplo, os valores de renda familiar são muito maiores que os de idade ou conceito do curso. O que se quer evitar é que o algoritmo considere a renda como atributo mais importante que os demais, porque as diferenças entre os registros para este atributo são significativamente maiores que as diferenças para os outros atributos.

```
scaler_fies = StandardScaler(with_mean=False)

fies_scaled = scaler_fies.fit_transform(fies_ohe_encoded)
```

7. Criação de Modelos de Machine Learning

7.1. Agrupamento Hierárquico

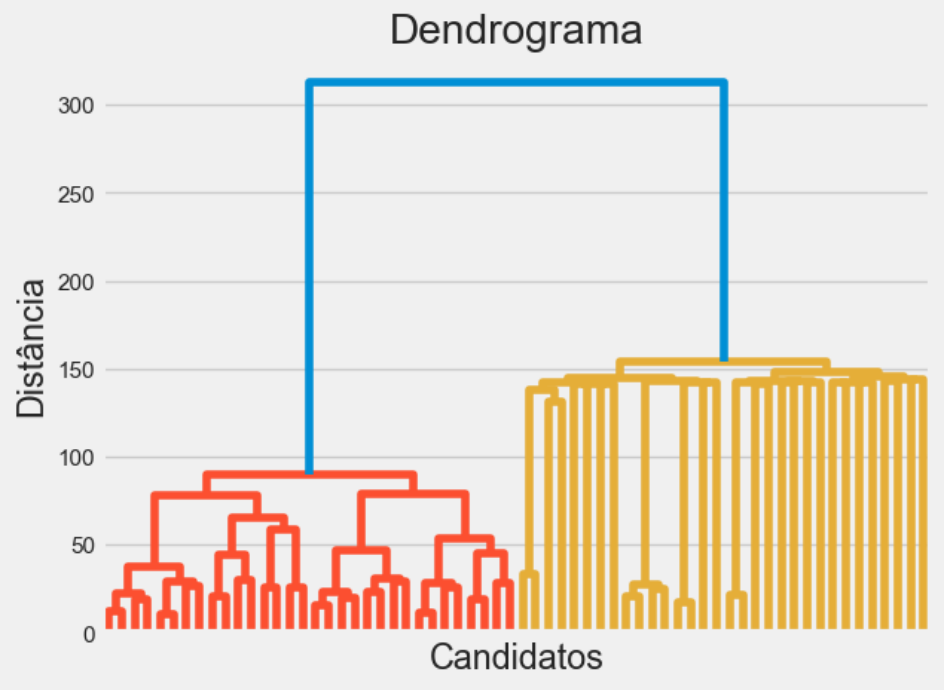
O agrupamento hierárquico consiste em agrupar objetos em clusters ou grupos com base em suas similaridades ou distâncias entre eles. Seu objetivo é criar uma hierarquia de clusters que possa ser visualizada como uma árvore (dendrograma), na qual a raiz representa todos os objetos, e as folhas representam cada objeto em seu próprio cluster. O processo de agrupamento hierárquico pode ser dividido em dois tipos: aglomerativo (ou botton-up) e divisivo (top-down).

No método aglomerativo se começa considerando cada objeto em seu próprio cluster. Se calcula a distância entre todos os pares de clusters. Se une os dois clusters mais próximos entre si em um único cluster. Se recalcula as distâncias entre o novo cluster e todos os outros clusters. Estes passos são repetidos até que todos os objetos estejam em um único cluster.

No método divisivo se começa com todos os objetos em um único cluster. Se calcula a distância entre todos os pares de objetos. Se divide o cluster em dois clusters, de forma que a distância entre os objetos dentro de cada cluster seja menor do que a distância entre os objetos em clusters diferentes. Se recalcula as distâncias entre os novos clusters e os outros objetos. Estes passos são repetidos até que cada objeto esteja em seu próprio cluster.

O dendrograma que é criado durante o processo de agrupamento hierárquico pode ser usado para visualizar a hierarquia de clusters e para determinar o número ideal de clusters que deve ser usado. O seguinte método foi usado para criar o dendograma. Para melhor visualização, ele foi limitado até o quinto nível.

```
dendrogram(linkage(fies_lab_encoded, method='ward'),
                           truncate_mode = 'level',
                           p = 5,
                           show_leaf_counts = False,
                           no_labels = True)
```



Dendograma resultado do agrupamento hierárquico dos dados.

Uma diretriz para encontrar a quantidade ideal de clusters é dada pela maior linha vertical no dendograma que não é cortada por nenhuma linha horizontal do diagrama. Neste caso a quantidade de clusters ideal é 2. O código seguinte corresponde à aplicação do agrupamento via método aglomerativo.

```
hc_fies = AgglomerativeClustering(n_clusters=2, affinity='euclidean',
linkage = 'ward')
hc_rotulos = hc_fies.fit_predict(fies_scaled.toarray())
```

Como resultado, os dados foram divididos em dois grupos com a seguinte quantidade de elementos: 7.579 (grupo 0) e 2.421 (grupo 1), identificando-se as seguintes diferenças do grupo 1 para o grupo 0:

- Em tamanho possui aproximadamente 1/3 dos candidatos do grupo 0
- O número de membros do grupo familiar é maior e superior a 2
- É mais jovem 1,5 anos
- A renda per capita é ~50% maior
- Almeja curso com nota de corte ~30% maior
- Possui notas mais altas em todas as disciplinas do ENEM

- Candidata-se com mais frequência a cursos ofertados mais longe de sua residência (~120 km em média)
- Está na região sudeste, enquanto o do grupo 0 na região nordeste
- Não cursou o ensino médio no ensino público
- Opta por curso de turno integral, enquanto no grupo 0 a opção é turno noturno
- Opta pelo curso de medicina, enquanto o grupo 0 pelo de direito
- É de cor branca, enquanto o do grupo 0 é de cor parda

Como característica em comum, os candidatos em ambos os grupos são do sexo feminino, concluíram o ensino médio no mesmo ano, optam por curso com o mesmo conceito no MEC.

Pode-se dizer que o grupo 1 possui condições socioeconômicas mais favoráveis que o grupo 0.

7.2. Agrupamento via K-Means

O algoritmo k-means tem por objetivo dividir um conjunto de dados em k clusters, de forma que os pontos dentro de cada cluster sejam o mais similares possível, enquanto os pontos em clusters diferentes sejam o mais diferentes possível. Seu funcionamento consiste das seguintes etapas.

1. Seleção do número de clusters (k): é necessário especificar o número de clusters que se deseja gerar a partir dos dados. É uma escolha crítica, pois determina como os dados serão agrupados. Mais adiante são abordados dois métodos para encontrar a quantidade ideal de clusters.
2. Inicialização dos centróides: O algoritmo seleciona aleatoriamente k objetos do conjunto de dados para atuar como centróides iniciais dos k clusters.
3. Atribuição dos objetos aos clusters: Cada objeto é atribuído ao cluster mais próximo ao seu centróide, com base na distância Euclidiana entre o objeto e o centróide.
4. Recálculo dos centróides: O centróide de cada cluster é recalculado para ser o centro de massa dos objetos atribuídos a esse cluster.

5. Repetição dos passos 3 e 4 até que não haja mais mudanças na atribuição dos objetos aos clusters. Isso significa que o algoritmo convergiu e os clusters são considerados estáveis.

O resultado do algoritmo k-means é uma partição dos dados em k clusters, onde cada objeto pertence a um cluster e cada cluster é definido por seu centróide.

Para identificar o número ideal de clusters dos dados analisados, aplicou-se primeiro o método do cotovelo, cuja ideia é analisar a relação entre a variância explicada (também conhecida como "inércia") e o número de clusters. A inércia mede a soma das distâncias quadradas de cada objeto ao centróide do cluster ao qual ele pertence. Quanto menor a inércia, mais compacto e homogêneo é um cluster. O objetivo é encontrar um ponto de inflexão (o "cotovelo") no gráfico de inércia em relação ao número de clusters, o que indica o número ideal de clusters a ser usado na análise. O código seguinte implementa essa ideia e produziu o gráfico na sequência.

```
sse = []
k_range = range(1, 11)
for k in k_range:
    kmeans = KMeans(n_clusters=k, init = 'k-means++', random_state = 0)
    kmeans.fit(fies_scaled)
    sse.append(kmeans.inertia_)
plt.style.use("fivethirtyeight")
plt.plot(k_range, sse)
plt.xticks(k_range)
plt.xlabel("Número de Clusters")
plt.ylabel("SSE")
plt.title('Método do Cotovelo para o K-Means')
plt.show()
```

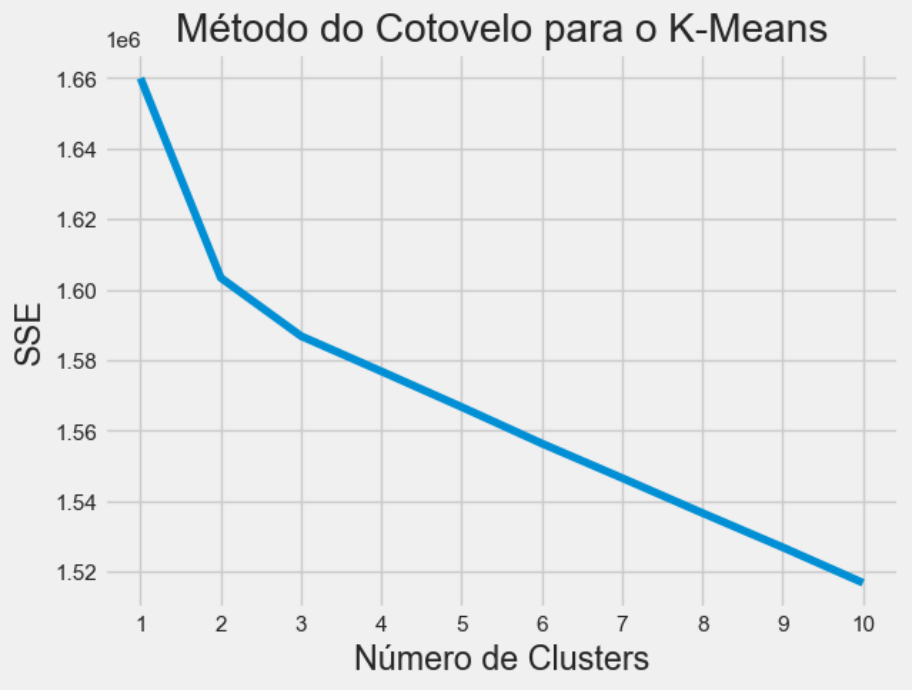


Gráfico resultante do método do cotovelo: o ponto de inflexão é sse=2.

Aplicou-se então o algoritmo com este parâmetro conforme o código a seguir.

```
kmeans = KMeans(n_clusters=2, init = 'k-means++', random_state = 0)
kmeans_rotulos = kmeans.fit_predict(fies_scaled)
np.unique(kmeans_rotulos, return_counts=True)
```

Como resultado, os dados foram divididos em dois grupos com a seguinte quantidade de elementos: 2.929 (grupo 0) e 7.071 (grupo 1), identificando-se as seguintes diferenças do grupo 0 para o grupo 1:

- Tem tamanho de ~40% dos candidatos do grupo 1
- O número de membros do grupo familiar é ligeiramente maior, superior a 2
- É mais jovem 1,5 anos
- A renda per capita é quase 60% maior
- Almeja curso com nota de corte ~30% maior
- Possui notas mais altas em todas as disciplinas do ENEM
- Candidata-se com mais frequência a cursos ofertados mais longe de sua residência (~100 km)
- Está na região sudeste, enquanto o do grupo 0 na região nordeste
- Não cursou o ensino médio no ensino público

- Opta por curso de turno integral, enquanto no grupo 1 a opção é turno noturno
- Opta pelo curso de medicina, enquanto o grupo 1 pelo de direito
- É de cor branca, enquanto do grupo 1 é de cor parda

Como característica em comum, os candidatos em ambos os grupos são do sexo feminino, concluíram o ensino médio no mesmo ano e concentram-se na região sudeste (a mais populosa). Em termos práticos, a classificação ficou parecida com a do algoritmo de agrupamento hierárquico.

Pode-se dizer que o grupo 0 possui condições socioeconômicas mais favoráveis que o grupo 1.

7.3. Agrupamento via PAM (K-medoids)

O algoritmo Partitioning Around Medoids (PAM) é um algoritmo de agrupamento que busca dividir um conjunto de dados em k clusters, onde k é um número pré-definido pelo usuário. O PAM é uma versão do algoritmo K-medoids, onde os centróides são substituídos por medóides.

O PAM começa selecionando aleatoriamente k medóides dos dados de entrada. Em seguida, ele atribui cada ponto de dados ao medóide mais próximo e calcula a soma das distâncias dos pontos de dados para seus medóides atribuídos. Esta soma é conhecida como a soma das distâncias ao quadrado (SSQ).

O algoritmo então tenta melhorar a qualidade do agrupamento, iterativamente substituindo um medóide por um ponto não selecionado aleatoriamente e recalculando a SSQ. Se a nova configuração de medóides resultar em uma redução na SSQ, a nova configuração é mantida. Esse processo é repetido até que não seja possível melhorar mais a qualidade do agrupamento.

Uma das principais vantagens do PAM é que ele funciona bem em conjuntos de dados com alta dimensionalidade ou com dados categóricos. Além disso, como o PAM usa

medóides em vez de centróides, ele é mais robusto em relação a valores atípicos do que o algoritmo K-means.

Assim como no K-Means, o parâmetro principal a informar ao algoritmo é o número de clusters. E novamente, para identificar o número ideal de clusters dos dados, aplicou-se também o método do cotovelo. Isto envolve executar o algoritmo para diferentes valores de k (número de clusters) e plotar o resultado em um gráfico. No gráfico, o eixo x representa o número de clusters (k), e o eixo y representa a medida de qualidade do agrupamento. Em seguida, deve-se procurar ponto de inflexão (ou "cotovelo") no gráfico, que é o ponto onde a melhoria na qualidade do agrupamento começa a diminuir significativamente à medida que k aumenta. O código seguinte implementa o método e produziu o gráfico na sequência.

```
sse = []
k_range = range(1, 11)
for k in k_range:
    pam = KMedoids(n_clusters=k)
    pam.fit(fies_scaled)
    sse.append(pam.inertia_)
plt.style.use("fivethirtyeight")
plt.plot(k_range, sse)
plt.xticks(k_range)
plt.xlabel("Número de Clusters")
plt.ylabel("SSE")
plt.title('Método do Cotovelo para o PAM')
plt.show()
```

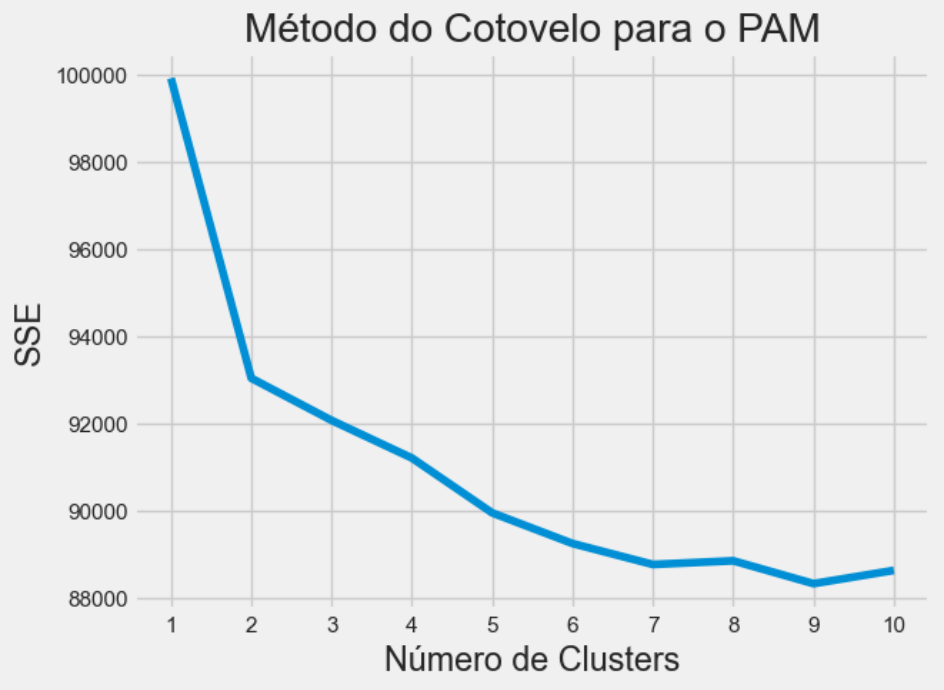


Gráfico resultante do método do cotovelo: o ponto de inflexão é sse=2.

Como o método do cotovelo sugeriu 2 clusters, aplicou-se o algoritmo com este parâmetro conforme o código a seguir.

```
pam = KMedoids(n_clusters=2).fit(fies_scaled)
np.unique(pam.labels_, return_counts=True)
```

Como resultado, os dados foram divididos em três grupos com a seguinte quantidade de elementos: 6.600 (grupo 0) e 3.400 (grupo 1), identificando-se as seguintes diferenças do grupo 1 para o grupo 0:

- Em tamanho possui aproximadamente 1/2 dos candidatos do grupo 0
- O número de membros do grupo familiar é maior e superior a 2
- É mais jovem ~1,5 anos
- A renda per capita é quase 60% maior
- Almeja curso com nota de corte ~25% maior
- Possui notas mais altas em todas as disciplinas do ENEM
- Candidata-se com mais frequência a cursos ofertados mais longe de sua residência (~80 km)
- Está na região sudeste, enquanto o do grupo 0 na região nordeste
- Não cursou o ensino médio no ensino público

- Opta por curso de turno integral, enquanto no grupo 0 a opção é turno noturno
- Opta pelo curso de medicina, enquanto o grupo 0 pelo de direito
- É de cor branca, enquanto o do grupo 0 é de cor parda

Como característica em comum, os candidatos em ambos os grupos são do sexo feminino, concluíram o ensino médio no mesmo ano, optam por curso com o mesmo conceito no MEC.

Pode-se dizer que o grupo 1 possui condições socioeconômicas mais favoráveis que o grupo 0. Em termos práticos, a classificação ficou parecida com a dos algoritmos anteriores.

7.4. Agrupamento via DBScan

O DBSCAN (Density-Based Spatial Clustering of Applications with Noise) é um algoritmo de clustering utilizado para agrupar pontos em um espaço n-dimensional, com base em sua densidade. Ele é capaz de encontrar clusters de formas arbitrárias e é eficiente em dados com ruído e outliers. Outra diferença quando comparado ao K-Means e ao PAM é que a quantidade de clusters é determinada pelo próprio algoritmo, que funciona da seguinte maneira:

1. Escolhe-se um ponto aleatório do conjunto de dados que ainda não tenha sido visitado.
2. Determina-se todos os pontos em seu "vizinho-epsilon" (também chamado de "vizinhança de densidade") que têm uma densidade suficiente (número mínimo de pontos em sua vizinhança-epsilon). Esses pontos são chamados de "pontos centrais".
3. Expansão do cluster: adicione o ponto central atual e seus pontos vizinhos (também pontos centrais) ao cluster atual. Repita este processo para cada ponto central adicionado.
4. Escolhe-se um novo ponto aleatório do conjunto de dados que ainda não tenha sido visitado e se repete o processo para formar um novo cluster.

5. Repete-se todo o processo para todos os pontos restantes até que todos os pontos tenham sido visitados.

Os dois parâmetros principais no DBSCAN são:

- Epsilon (eps): O tamanho do "vizinho-epsilon". Pontos que estão dentro do vizinho-epsilon são considerados próximos o suficiente para serem incluídos no mesmo cluster. Define a distância máxima entre dois pontos para que eles possam ser considerados parte do mesmo cluster. Um valor muito pequeno pode resultar em muitos clusters e um valor muito grande pode resultar em todos os pontos serem agrupados em um único cluster.
- Mínimo de pontos (min_samples): O número mínimo de pontos em uma vizinhança-epsilon para um ponto ser considerado um ponto central. Um valor muito baixo pode resultar em ruído sendo considerado como um cluster e um valor muito alto pode resultar em clusters muito pequenos.

Além disso o DBSCAN não classifica o que se chama ponto de ruído; um ponto que não é um ponto central e não está em uma vizinhança-epsilon de um ponto central.

Uma maneira de encontrar o valor ideal para o parâmetro eps é traçar um gráfico da distância média para os k vizinhos mais próximos (k-distance plot) e encontrar o valor de eps onde há um ponto de inflexão na curva da distância. Ou seja, de novo usou-se o método do cotovelo, com o seguinte código e que produziu o gráfico na sequência.

```
k = 4
nbrs = NearestNeighbors(n_neighbors=k+1).fit(fies_scaled)
distances, indices = nbrs.kneighbors(fies_scaled)
k_dist = distances[:, -1]

# Ordene as distâncias em ordem crescente
sorted_distances = np.sort(k_dist)

plt.plot(np.arange(len(sorted_distances)), sorted_distances)
plt.ylim(0,9)
plt.xlabel('Índice do ponto')
plt.ylabel('eps')
plt.title('Método do cotovelo para achar o valor ideal de eps')
plt.show()
```

Gráfico do método do cotovelo: o ponto de inflexão é entre 5 e 7.

Pelo método do cotovelo o valor ideal de eps está entre 5 e 7. Porém resta ainda achar o valor ideal para min_samples. A alternativa utilizada foi a técnica de busca em grade (grid search). Nessa técnica, uma grade de valores é definida para cada parâmetro a ser ajustado e o algoritmo é executado em cada combinação possível de valores dos parâmetros, sendo avaliado através de uma métrica de desempenho pré-definida, neste caso o score da silhueta.

Definindo a grade de valores para os parâmetros

```
param_grid = {'eps': np.arange(5, 8, 0.2), 'min_samples': np.arange(2, 10, 1)}

dbscan = DBSCAN()
best_score_silhouette = -1
best_eps_silhouette = 0
best_min_samples_silhouette = 0
for eps in param_grid['eps']:
    for min_samples in param_grid['min_samples']:
        dbscan.set_params(eps=eps, min_samples=min_samples)
        dbscan.fit(fies_scaled)
        if len(set(dbscan.labels_)) > 1:
            score_silhouette= silhouette_score(fies_scaled, dbscan.labels_)
            if score_silhouette > best_score_silhouette:
```

```
                best_score_silhouette = score_silhouette
                best_eps_silhouette = eps
                best_min_samples_silhouette = min_samples
```

Testou-se então o parâmetro eps no intervalo de 5 a 8, variando 0,2 a cada iteração e o parâmetro min_samples no intervalo de 2 a 10, variando 1 a cada iteração. Na prática, um teste de força bruta. O melhor score de silhueta foi alcançado com os valores eps=7,2 e min_samples=2.

Aplicou-se então o algoritmo com os parâmetros sugeridos conforme o código a seguir.

```
dbscan_fies = DBSCAN(eps = best_eps_silhouette,
       min_samples= best_min_samples_silhouette)
dbscan_fies.fit(fies_scaled)
np.unique(dbscan_fies.labels_, return_counts=True)
```

Como resultado, os dados foram divididos em 86 clusters, deixando 95 candidatos classificados como ruído. Muitos cluster com poucos candidatos (menos de 10), o que não tem muita utilidade. E um único cluster concentrou a maioria dos candidatos, cujo perfil não é muito diferente do perfil médio da amostra, om exceção da opção de curso do curso que é engenharia elétrica, enquanto na amostra é medicina.

6. Interpretação dos Resultados

O primeiro achado foi citado na seção de exploração de dados: os candidatos residem em 4.382 municípios e buscam cursos ofertados em 514 municípios. O que indica uma concentração na oferta de cursos em poucos municípios. E que poderia servir de base para o comitê gestor do FIES orientar junto com as instituições de ensino a ampliação na oferta de municípios atendidos pelo programa.

Outro achado foi que para mais da metade dos candidatos (57.310) a busca é por um curso em município diferente do de sua residência. Considerando que o programa é direcionado a um público de baixa renda, percebe-se mais uma barreira econômica para o acesso à graduação, além do valor da mensalidade, há o custo de transporte (ou até mudança) para outro município.

Uma alternativa a esse problema seria também a ampliação da oferta de cursos à distância, nos cursos onde isso é possível. Embora o FIES contemple cursos à distância, não foi possível identificar nos dados publicados da inscrição ao FIES se o curso selecionado pelo candidato era da modalidade presencial ou não. Esta informação também impacta no resultado da análise de agrupamento e seria uma melhoria relevante aos dados publicados.

Outra questão interessante observada é o lapso temporal entre a conclusão do ensino médio e sua tentativa de acesso ao FIES (de 3 a 6 anos), que merece ser investigada as razões. Será que o candidato prioriza a tentativa de acesso à universidade pública por um tempo para só então tentar o FIES? Será que o candidato não considera o curso superior imediatamente após a conclusão do ensino médio? Por que a nota do ENEM usada é a de 3 anos anteriores à oferta do FIES?

Sobre os resultados da aplicação dos 4 algoritmos de agrupamento, percebe-se a divisão de clusters nos três primeiros por características predominantemente socioeconômicas. Isto talvez pelo fato de haver forte correlação e/ou causalidade entre as variáveis disponíveis. Por exemplo, o candidato com grupo familiar maior, tem renda per capita e notas mais altas. A nota é explicada pela renda per capita? E esta por sua vez é explicada pelo tamanho do grupo familiar? Esta investigação foge ao escopo deste trabalho.

Um fato que chama a atenção nos resultados é a predominância do candidato de cor branca almejando o curso de medicina em instituição particular. A mensalidade deste curso é a mais cara, em média de quase dez salários-mínimos, o que faz com que o curso seja elitista. Mesmo que o candidato consiga obter financiamento e em um percentual alto, ainda assim seria um valor incompatível com a renda declarada. O que permite supor que talvez o FIES seja uma alternativa tentada por candidatos de classe média, que de certa forma se inscrevem no programa tentando burlar aos critérios estabelecidos. Este tema também merece uma investigação à parte, podendo ter tratado em trabalho futuro.

Não se alcançou o resultado desejado no agrupamento pelo algoritmo DBScan. Não se conseguiu obter uma configuração adequada dos seus parâmetros, mesmo com a estratégia do cotovelo e grid search. Tentou-se também empiricamente outros valores, porém valores baixos de eps resultaram em muito ruído na classificação. E valores mais altos de min_samples (conjugado com eps alto), também resultou em alto nível de ruído (mas menos quando comparado ao eps baixo). Chegou-se ao dilema de poucos clusters e um ruído alto ou ruído baixo e dezenas de clusters.

Outra explicação para o resultado do DBScan pode ser devido à alta dimensionalidade dos dados, mesmo com a redução de dimensões feita no pré-processamento. Isso ocorre pois o algoritmo é adequado para dados com alta densidade, e esta pode ser diluída em alta dimensionalidade (principalmente após o uso do One Hot Encoder), tornando difícil identificar grupos significativos.

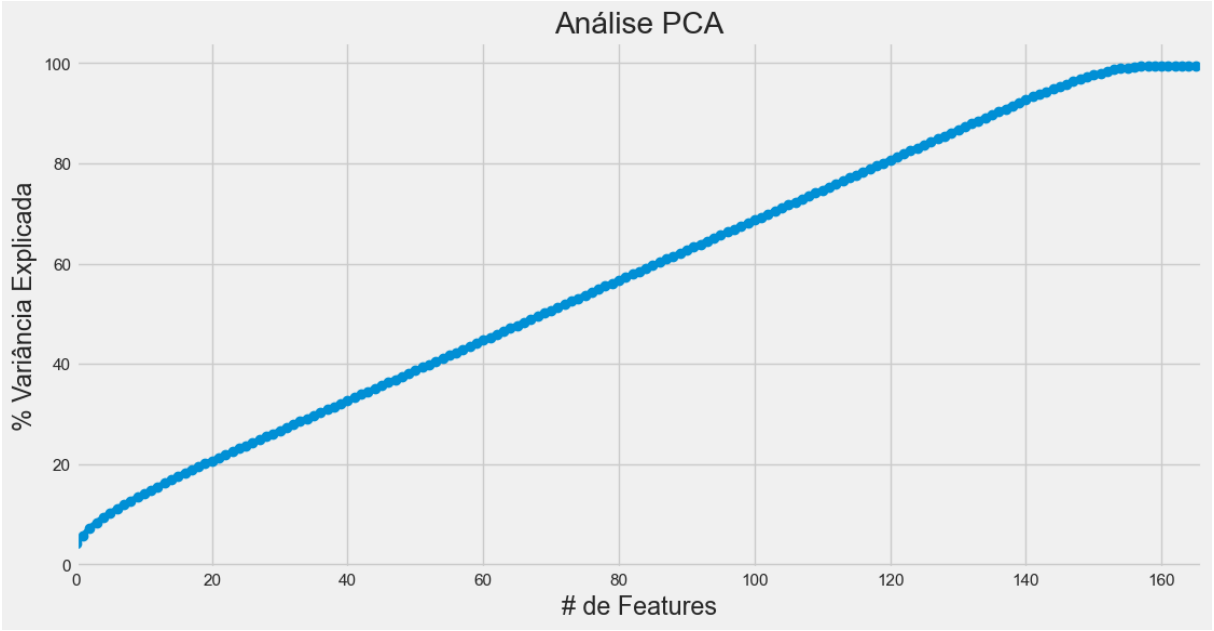
Tentou-se a redução de dimensionalidade via PCA. O PCA (Principal Component Analysis) é uma técnica de análise multivariada que permite reduzir a dimensionalidade de um conjunto de dados, mantendo a maior parte da variação presente nos dados originais. Ele ajuda a identificar as principais características ou componentes que contribuem para a variação dos dados. Funciona transformando os dados originais em um conjunto de novas variáveis ortogonais, chamadas de componentes principais, que são classificadas em ordem decrescente de importância. Os componentes principais capturam a maior quantidade possível de variação nos dados originais e podem ser usados para visualização do resultado do agrupamento.

A escolha das novas dimensões tem relação com o quanto do % de variância se deseja manter, isso determina qual a quantidade de componentes (ou dimensões) deve ser usada. Para se manter 100% de variância explicada, não há redução alguma de dimensionalidade. Então é necessário renunciar à precisão nos resultados para conseguir a redução de dimensão. Não há uma regra para determina o % de variância que se deseja manter. Mas tipicamente se seleciona de 80% a 90% de variância. O método do cotovelo também pode ser aplicado a este caso.

O código seguinte aplica o PCA nos dados e produziu o gráfico exibido na sequência.

```
pca = PCA(n_components=fies_scaled.shape[1])

pca.fit(fies_scaled.toarray())

variance = pca.explained_variance_ratio_

var = np.cumsum(np.round(variance, 3)*100)

plt.figure(figsize=(12,6))

plt.ylabel('% Variância Explicada')

plt.xlabel('# de Features')

plt.title('Análise PCA')
```

```
plt.xlim(0,fies_scaled.shape[1])

plt.plot(var, marker = 'o')
```



% de variância explicada sobre a base de dados do FIES com 166 variáveis.

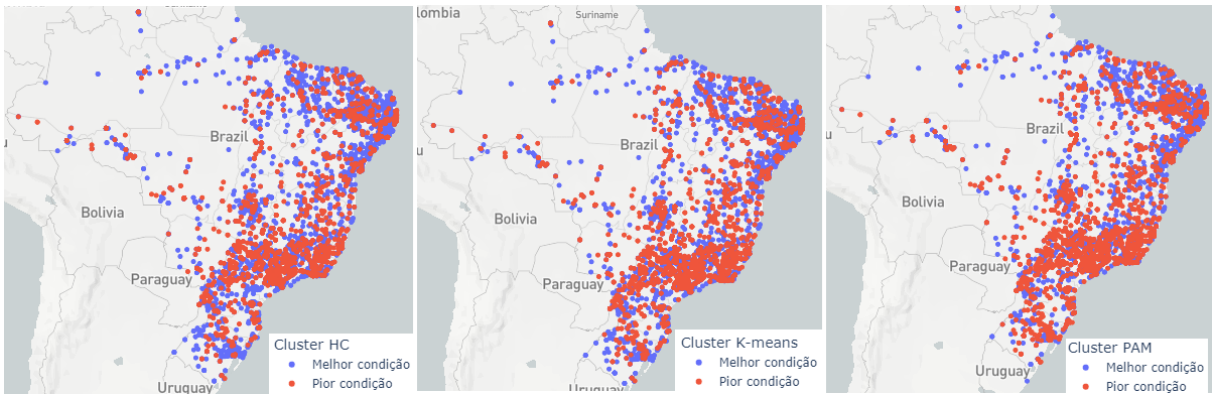
Observando o gráfico, percebe-se que para um % de variância acima de 80% a redução de dimensões ainda é pequena, restam mais de 100 dimensões. O gráfico também não apresenta nenhum ponto de inflexão. Isso pode indicar alta correlação entre as variáveis.

7. Apresentação dos Resultados

Durante o desenvolvimento deste trabalho surgiram questões que não puderam ser respondidas face a ausência de dados; o que sugere que mais atributos que poderiam ser publicados sobre a inscrição do FIES, ou até mesmo passarem a ser coletados, caso não existam. Por exemplo:

- valor da mensalidade do curso
- duração do curso
- se o candidato está cursando um curso superior
- se o candidato já evadiu de um curso superior e o motivo
- há quantos anos ele tenta ingressar no ensino superior (ou quantas vezes ele prestou o ENEM, é um dado que já existe e pode cumprir o mesmo propósito)
- a modalidade do curso (presencial, semipresencial, à distância)

Os resultados dos três primeiros algoritmos foram similares. A seguir são ilustrados no mapa o agrupamento de cada algoritmo conforme a residência do candidato.



8. Links

A seguir estão os links para o vídeo com a apresentação deste trabalho e para o repositório contendo os dados utilizados no projeto, scripts do Jupyter Notebook e a monografia.

Link para o vídeo: https://youtu.be/32yblVSAzs8

Link para o repositório: https://github.com/gssimoes/TCC-PUCMG-CienciaDados

APÊNDICE

```
#Instala o pacote
get_ipython().system('pip install scikit-learn-extra')


#Importa as bibliotecas a serem utilizadas
import pandas as pd
import numpy as np
from datetime import datetime as dt
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from unidecode import unidecode
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.neighbors import NearestNeighbors
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn_extra.cluster import KMedoids


token = ""
px.set_mapbox_access_token(token)


#Define pasta e arquivos de trabalho
pasta = 'D:/Ciência de Dados/16. TCC/Trabalho/'
arq_fies = 'relatorio_inscricao_dados_abertos_fies_22020.csv'
arq_latlong_mun = 'municipios.csv'
arq_stn_mun = 'TABMUN.csv'


#Carrega dados de inscrições do FIES
#A separação dos campos foi feita usando ponto e vírgula
#A codificação de caracteres do arquivo é Windows 1252
#Valores numéricos estão no formato brasileiro, com separador de milhar com
ponto e fração com vírgula
df_fies = pd.read_csv(pasta + arq_fies, sep = ';', encoding='cp1252', deci-
mal=',', thousands='.')
```

```
df_fies.info()


#Carrega dados de latitude e longitude dos municípios
#A separação dos campos foi feita usando vírgula
df_latlong_mun = pd.read_csv(pasta + arq_latlong_mun, sep = ',',
usecols=[0,1,2,3,6])
df_latlong_mun.info()


#Carrega dados de código SIAFI e IBGE dos municípios
#A separação dos campos foi feita usando ponto e vírgula
#O arquivo foi disponibilizado sem cabeçalho, a primeira linha deve ser trata-
da como dado
#A segunda coluna (significado desconhecido) não será carregada
#Será criado um cabeçalho com o nome das colunas no df
df_stn_mun  = pd.read_csv(pasta + arq_stn_mun, sep = ';', header=None,
usecols=[0,2,3,4],
                          names=['siafi_id', 'nome_mun', 'sigla_uf', 'codi-
go_ibge'])
df_stn_mun.info()


# # Pré-processamento dos dados

# ## Remoção de dados duplicados

df_latlong_mun.drop_duplicates(keep='first', inplace=True)
#Executa-se o info de novo para ver se a quantidade de linhas continua igual
(sem duplicidade neste caso)
df_latlong_mun.info()


df_stn_mun.drop_duplicates(keep='first', inplace=True)
#Executa-se o info de novo para ver se a quantidade de linhas continua igual
(sem duplicidade neste caso)
df_stn_mun.info()


df_fies.drop_duplicates(keep='first', inplace=True)
#Executa-se o info de novo para ver se a quantidade de linhas continua igual
(2902 linhas repetidas eliminadas!)
df_fies.info()


#Remove registros que não sejam da primeira opção de curso do candidato
df_fies.drop(df_fies[df_fies['Opções de cursos da inscrição'] != 1].index,
inplace=True)
#Foram removidos 101.610 registros
```

```
df_fies.shape


# ## Seleção inicial de atributos de inscrições do FIES

#Remove a coluna de opções pois restaram apenas os registros relativos à opção
1
del df_fies['Opções de cursos da inscrição']

#Ano e semestre são constantes em toda base, logo irrelevantes para o algorit-
mo de ML
del df_fies['Ano do processo seletivo']
del df_fies['Semestre do processo seletivo']

#ID do estudante não é atributo relevante para o algoritmo de ML
del df_fies['ID do estudante']

#Como a intenção é analisar o perfil dos candidatos do FIES, os atributos re-
lativos à situação do financiamento obtido
#serão descartados pois os candidatos que não obtiveram o FIES não possuem
dados nestas colunas
del df_fies['Situação Inscrição Fies']
del df_fies['Percentual de financiamento']
del df_fies['Semestre do financiamento']
del df_fies['Qtde semestre financiado']

#Remove colunas relativos aos dados da IES (Instituição de Ensino Superior)
ofertante do curso,
#mantendo apenas município onde o curso é ofertado
#A premissa é de que a opção do candidato por uma IES tem haver com a oferta
do curso desejado no município de interesse e
#com o valor da mensalidade (embora este atributo não exista na base de dados)
del df_fies['Nome mantenedora']
del df_fies['Natureza Jurídica Mantenedora']
del df_fies['CNPJ da mantenedora']
del df_fies['Código e-MEC da Mantenedora']
del df_fies['Nome da IES']
del df_fies['Código e-MEC da IES']
del df_fies['Organização Acadêmica da IES']
del df_fies['Município da IES']
del df_fies['UF da IES']
del df_fies['Nome do Local de oferta']
del df_fies['Código do Local de Oferta']

#Estes atributos são definidos pelo IBGE para agrupar municípios, levando em
conta fatores como proximidade geográfica,
#características econômicas, sociais e culturais em comum.
#Isto é utilizado para a formação de grupos de preferência, que priorizam es-
tudantes que residem em regiões com baixo IDHM.
```

```
#Considerou-se que há redundância destes atributos com o município e UF de
residência do candidato.
del df_fies['Região grupo de preferência']
del df_fies['UF']
del df_fies['Microrregião']
del df_fies['Cod.Microrregião']
del df_fies['Mesorregião']
del df_fies['Cod.Mesorregião']

#Estes atributos identificam a área temática do curso de graduação do candida-
to, a classificação é definida pela CAPES.
#Como será utilizado o curso do candidato no algoritmo de classificação, op-
tou-se por eliminar estes atributos.
#Eles podem introduzir um viés no algoritmo de classificação pois reduzem o
peso da escolha do curso pelo candidato.
#Por exemplo, cursos diferentes mas de mesma área e subárea tendem a aproximar
candidatos (viés da CAPES).
del df_fies['Área do conhecimento']
del df_fies['Subárea do conhecimento']

#O grupo de preferência é derivado do estado, município e o curso que o candi-
dato quer estudar.
#Por ser um atributo derivado de outros, é redundante.
del df_fies['Cod. do Grupo de preferência']


#Avalia se código do curso e nome do curso são colunas equivalentes.
df_fies[['Código do curso','Nome do curso']].nunique()
#Os valores únicos estão divergentes, há vários códigos correspondentes a um
mesmo nome de curso.


#Analisa amostra de registros do curso de direito
linhas_filtro_df = df_fies['Nome do curso'] == 'DIREITO'
df_fies[linhas_filtro_df]['Código do curso'].nunique()


#Não foi possível descobrir porque cursos de mesmo nome podem receber códigos
diferentes.
#Para favorecer a aproximação pelo algoritmo de candidatos com curso de mesmo
nome, mesmo com códigos diferentes,
#optou-se por manter o nome e eliminar o código do curso.
del df_fies['Código do curso']


# O conceito de curso é a nota de qualidade dada pelo MEC aos cursos de gradu-
ação de IES credenciadas.
# Poderão ser financiados os cursos de graduação com conceito maior ou igual a
3, numa escala de 1 a 5.
```

```
# Os cursos que ainda não possuam avaliação mas que estejam autorizados para
funcionamento, poderão participar do FIES.
# Estes cursos aparecem com o conceito 'Autorizado' e serão substituídos pelo
valor zero para transformar a coluna em numérica.

#Como essas colunas foram carregadas como tipo object, verifica valores dis-
tintos existentes.
df_fies[['Conceito de curso do GP','Conceito']].value_counts()

#Substitui valor 'Autorizado' por '0' e transforma o tipo das colunas para
numérico
df_fies['Conceito de curso do GP'] = df_fies['Conceito de curso do
GP'].replace('Autorizado', '0')
df_fies['Conceito de curso do GP'] = df_fies['Conceito de curso do
GP'].astype('int64')
df_fies['Conceito'] = df_fies['Conceito'].replace('Autorizado', '0')
df_fies['Conceito'] = df_fies['Conceito'].astype('int64')


#Analisa relação entre os dois atributos via gráfico de dispersão.
plt.xlabel("Conceito de curso do GP")
plt.ylabel("Conceito")
plt.scatter(x = df_fies['Conceito de curso do GP'], y = df_fies['Conceito'])
plt.show()

#Há coincidência de pontos em todas as ocorrências, bem como a correlação é
perfeita.
#Apenas um dos atributos é necessário para o algoritmo de clusterização. Se
elimina uma das colunas devido à redundância.
del df_fies['Conceito de curso do GP']


#Analisa valores únicos da coluna para tentar entender seu significado.
df_fies['Grau'].value_counts()


#Observa-se que 98% das ocorrências são para o valor semestral, o que indica
baixa variância dos valores da coluna.
#Tomando como premissa que esse dado não seria um fator relevante para a esco-
lha do curso pelo candidato;
#como seria por exemplo a duração do curso ou valor (atributos inexistentes na
base), optou-se por eliminar o atributo.
del df_fies['Grau']


# Verifica em quantas linhas há coincidência entre a média nota ENEM e a soma
das 5 notas individuais dividida por 5
```

```
# Há diferenças de centavos ocasionado por arredondamento em algumas linhas,
portanto será considerado que não há coincidência apenas quando a diferença
for superior a R$1
linhas_filtro_df = abs(df_fies['Média nota Enem'] - ((df_fies['Redação'] +
                                                      df_fies['Matemática e
suas Tecnologias'] +
                                                      df_fies['Linguagens,
Códigos e suas Tec'] +
                                                      df_fies['Ciências Natu-
reza e suas Tec'] +
                                                      df_fies['Ciências Huma-
nas e suas Tec']) / 5)) <= 1
df_fies[linhas_filtro_df].describe()


#Como para todas as linhas se confirmou que a coluna Média nota Enem é a média
aritmética das outras 5 notas
#E também que não há nenhuma nota média de ENEM inferior a 450 (nota mínima
exigida pelo FIES)
#Será eliminada a coluna, pois é redundante
del df_fies['Média nota Enem']


# # Transformação de dados

#Observa formato carregado da data de nascimento. O ano possui apenas dois
dígitos, porém há datas em séculos diferentes.
df_fies['Data de Nascimento'].head

#Converte a data de nascimento para o tipo datetime64 em vez de object
df_fies['Data de Nascimento'] =  pd.to_datetime(df_fies['Data de Nascimento'],
infer_datetime_format=True)
#Mostra as datas de nascimento ordenadas pelas mais recentes; há datas com
anos maiores que 2020.
#Ocorreu o Bug do Milênio ao converter a string para datetime64
df_fies.sort_values('Data de Nascimento', ascending=False)['Data de Nascimen-
to'].head

# Para corrigir o bug do milênio na conversão de data do ano com 2 dígitos,
foram subtraídos 100 anos das datas de
# nascimento superiores ao ano da oferta do FIES (2020)
cutoff_date = pd.to_datetime('01-01-2020')
df_fies.loc[df_fies['Data de Nascimento'] > cutoff_date, 'Data de Nascimento']
-= pd.DateOffset(years=100)
# lista novamente as datas ordenadas para certificar que não há nenhuma supe-
rior a 2020
df_fies.sort_values('Data de Nascimento', ascending=False)['Data de Nascimen-
to'].head
```

```
#Remove registros em que a data de nascimento é maior ou igual ao ano de con-
clusão do ensino médio (neste caso há erro nos dados)
df_fies.drop(df_fies[df_fies['Data de Nascimento'].dt.year >= df_fies['Ano
conclusão ensino médio']].index, inplace=True)
len(df_fies.index)
#Apenas 1 registro removido

#Remove registros cuja ano de nascimento seja maior ou igual ao ano do ENEM
(neste caso há erro nos dados)
df_fies.drop(df_fies[df_fies['Data de Nascimento'].dt.year >= df_fies['Ano do
Enem']].index, inplace=True)
len(df_fies.index)
#Nenhum registro foi removido

#Calcula a idade do candidato em anos quando a inscrição ao FIES 2020/2 foi
aberta (09/08/2020)
df_fies['Idade'] = ((pd.to_datetime('09/08/2020', dayfirst=True) -
df_fies['Data de Nascimento'])/365).dt.days
#Uma vez que a idade já foi calculada, a Data de nascimento passa a ser redun-
dante e pode ser eliminada
del df_fies['Data de Nascimento']


# # Análise Exploratória
#
# Usou-se o método describe para obter uma visão descritiva das estatísticas
do DF em busca de algo estranho

df_latlong_mun.describe(include='all')
#Nada estranho.
#Latitude dentro da faixa [-90..90]
#Longitude dentro da faixa [-180..180].
#Os dados estatísticos das colunas numéricas do SIAFI e IBGE não são relevan-
tes pois os dados são categóricos nominais.
#Há duplicidade de nomes de municípios já era esperada, tendo em vista que há
municípios de mesmo nome em UF distintas.


df_stn_mun.describe(include='all')
#Nada estranho.
#Os dados estatísticos das colunas numéricas do SIAFI e IBGE não são relevan-
tes pois os dados são categóricos nominais.
#Há duplicidade de nomes de municípios já era esperada, tendo em vista que há
municípios de mesmo nome em UF distintas.

#Analisa primeiro as colunas numéricas.
df_fies.describe()
#Faixa de idade: [15..74], sem valores absurdos (ex.: <=0 ou >100)
```

```
#Faixa de ano para conclusão ensino médio: [1969..2020], sem valores absurdos (>2020, superior ao ano da oferta do FIES), o menor ano é do candidato com maior idade
#Faixa de número de membros do grupo familiar: [0..13], sem valores absurdos (ex.: <=0 ou >20)
#Faixa notas do ENEM, sem valores absurdos (ex.: <=450 ou >1.000)
#Faixa do ano do ENEM: [2010..2020], sem valores absurdos (>2020, superior ao ano da oferta do FIES)
#Não há regra entre o ano do ENEM que o candidato prestou o exame e o ano de conclusão do ensino médio. Treineiros podem fazer o ENEM antes de concluir o ensino médio.
#Há algo estranho para o número de membros do grupo familiar com valor zero
#Há algo estranho entre a quantidade de linhas da renda familiar e da renda per capita


# Sobre o  Nº de membros Grupo Familiar
# Não há valores nulos na coluna (a quantidade de valores da coluna é igual à quantidade de linhas do dataframe).
df_fies[df_fies['Nº de membros Grupo Familiar'] == 0].describe()
# Em todos estes casos a renda familiar é NAN, porém há valores na renda per capita.
# Isto indica que o MEC não considera o próprio candidato como parte do número de membros do grupo familiar

# Verificação em sentido inverso, de que para todas as linhas cuja renda familiar é NAN, o número de membros do grupo familiar é zero.
# Isso indica que a renda familiar mensal é informada apenas quando há outros membros no grupo familiar.
# Observa-se também que a renda per capita média não é zero, o que indica que a renda per capita computa a renda do candidato e a renda familiar não.
df_fies[df_fies['Renda familiar mensal bruta'].isna()].describe()

#Verifica o número de membros do grupo familiar quando a renda familiar é zero.
#Há 21 ocorrências, e em todas o número de membros familiares é maior que zero e renda per capita é zero.
#Isso indica que o candidato informou membros da família, maior de 14 anos e sem renda.
df_fies[df_fies['Renda familiar mensal bruta'] == 0].describe()


#Verifica em quantas linhas há coincidência entre a renda familiar e a renda per capita multiplicada pelo número de membros.
linhas_filtro_df = round(df_fies['Renda familiar mensal bruta'],2) == round(df_fies['Renda mensal bruta per capita'] * df_fies['Nº de membros Grupo Familiar'],2)
df_fies[linhas_filtro_df][['Nº de membros Grupo Familiar', 'Renda familiar mensal bruta', 'Renda mensal bruta per capita']].describe()
```

```
#A coincidência existe apenas quando ambas as colunas estão zeradas, o que
indica que o candidato não compõe o número de membros.

#Preenche a coluna renda familiar com zeros caso o valor seja nulo (neste caso
o número de membros do grupo já é zero)
df_fies['Renda familiar mensal bruta'].fillna(0, inplace=True)
#Verifica total de linhas cuja renda familiar é zero
#Deveria ser a soma dos NAN (27.077) e os de zero (21), confere!
df_fies[df_fies['Renda familiar mensal bruta'] == 0].describe()

#Verifica em quantos registros há renda familiar menor que a renda per capita.
linhas_filtro_df = round(df_fies['Renda familiar mensal bruta'],2) <
round(df_fies['Renda mensal bruta per capita'],2)
df_fies[linhas_filtro_df][['Nº de membros Grupo Familiar', 'Renda familiar
mensal bruta', 'Renda mensal bruta per capita']].describe()
#Em todas as ocorrências a renda familiar é zero, o que indica que a renda do
candidato não compõe a renda familiar.

#Verifica em quantas linhas há coincidência entre a renda familiar e a renda
percapita multiplicada pelo número de membros + 1 (o próprio candidato)
#Há diferenças de centavos ocasionado por arredondamento em algumas linhas,
portanto será considerado que não há coincidência apenas quando a diferença
for superior a R$1
linhas_filtro_df = abs(df_fies['Renda familiar mensal bruta'] - df_fies['Renda
mensal bruta per capita'] * (df_fies['Nº de membros Grupo Familiar']+1)) <= 1
df_fies[linhas_filtro_df][['Nº de membros Grupo Familiar', 'Renda familiar
mensal bruta', 'Renda mensal bruta per capita']].describe()

# Investiga a situação dos registros em que não há a coincidência anterior
linhas_filtro_df = abs(df_fies['Renda familiar mensal bruta'] - df_fies['Renda
mensal bruta per capita'] * (df_fies['Nº de membros Grupo Familiar']+1)) > 1
df_fies[linhas_filtro_df][['Nº de membros Grupo Familiar', 'Renda familiar
mensal bruta', 'Renda mensal bruta per capita']].describe()
# Em todas elas a renda familiar e o número de membros estão zerados
# Se consideramos a tamanho da família somente o próprio candidato, a renda
familiar seria equivalente à renda per capita
# logo a renda familiar é uma informação redundante, derivada do número de
membros e a renda per capita,
# podendo ser desprezada no modelo de machine learning.

# Renda familiar mensal bruta é dado redundante (renda per capita x número
membros família), atributo pode ser eliminado
del df_fies['Renda familiar mensal bruta']

#Reanalisa  as colunas numéricas após o processamento anterior.
df_fies.describe()
#O candidato médio do FIES tem 23 anos, concluiu o ensino médio em 2014, vive
com menos que dois familiares, cuja renda mensal per capita é
```

```
#de R$1.083, busca um curso de conceito entre 3 e 4, cuja nota de corte é 616,
sendo que sua nota do ENEM é do ano 2017 e de média 564.

#Analisa as colunas não numéricas.
df_fies.describe(include=[object])
#O candidato médio do FIES é do sexo feminino, reside em São Paulo-SP, é de
cor parda, não possui deficiência física,
#cursou o ensino médio em escola pública, nunca concluiu o curso superior nem
foi professor da rede pública de ensino, deseja
#realizar o curso de Medicina na própria cidade e o turno preferido é noturno.

df_fies['Sexo'].value_counts()
df_fies['UF de residência'].value_counts()
df_fies['Municipio de residência'].value_counts()
df_fies['Etnia/Cor'].value_counts()
df_fies['Pessoa com deficiência?'].value_counts()
df_fies['Tipo de escola no ensino médio'].value_counts()
df_fies['Concluiu curso superior?'].value_counts()
df_fies['Professor rede pública ensino?'].value_counts()
df_fies['Munícipio do Local de Oferta'].value_counts()
df_fies['UF do Local de Oferta'].value_counts()
df_fies['Nome do curso'].value_counts()
df_fies['Turno'].value_counts()

#Elimina acentos e caracteres especiais da coluna para poder fazer o merge com
dados de latitude/longitude do município pelo nome.
df_fies['Municipio de residência'] = df_fies['Municipio de residên-
cia'].apply(lambda x: unidecode(x))
df_fies['Munícipio do Local de Oferta'] = df_fies['Munícipio do Local de Ofer-
ta'].apply(lambda x: unidecode(x))
df_fies['Municipio de residência'].value_counts()


df_fies['Munícipio do Local de Oferta'].value_counts()

#O apóstofro existente no nome do município foi substituído na base original
por um caracter especial.
#Provavelmente houve problema na exportação dos dados pelo MEC. Substitui o
caracter especial pelo apóstofro.
df_fies['Municipio de residência'] = df_fies['Municipio de residên-
cia'].str.replace('\?', '\'')
df_fies['Munícipio do Local de Oferta'] = df_fies['Munícipio do Local de Ofer-
ta'].str.replace('\?', '\'')
df_fies['Municipio de residência'].value_counts()

#Analisa qual a incidência de casos onde o candidato seleciona um curso ofer-
tado num município diferente do de sua residência.
#Se a coincidência for alta, indica que um dos atributos poderia ser despreza-
do.
```

```
linhas_filtro_df = df_fies['Municipio de residência'] != df_fies['Munícipio do Local de Oferta']
df_fies[linhas_filtro_df][['Municipio de residência','Munícipio do Local de Oferta']].describe()
#Porém não é o que acontece, em mais da metade dos casos o candidato seleciona um curso em um município diferente
#do município de sua residência. Logo os dois atributos serão mantidos.


# # Enriquecimento dos dados
#Mescla dados de município latitude/longitude com lista siafie/ibge para obter a sigla UF
df_mun_merged = pd.merge(df_latlong_mun, df_stn_mun, on=['siafi_id', 'codigo_ibge'], how='inner')
df_mun_merged

#Elimina as colunas de código IBGE, id SIAFI e do nome com acentuação que não são mais necessárias.
#E transforma coluna do nome acentuado também para maiúscula, eliminando espaços antes e depois do nome.
del df_mun_merged['codigo_ibge']
del df_mun_merged['siafi_id']
del df_mun_merged['nome']
df_mun_merged['nome_mun'] = df_mun_merged['nome_mun'].str.strip().str.upper()
#Elimina acentos e caracteres especiais da coluna para poder fazer o merge com dados de lat long do município pelo nome
df_mun_merged['nome_mun'] = df_mun_merged['nome_mun'].apply(lambda x: unidecode(x))
df_mun_merged


# ## Enriquecendo dados de município de residência

#Renomeia colunas para fazer o merge com os dados de município de residencia do candidato do fies
df_mun_merged.rename(columns={'nome_mun': 'Municipio de residência', 'sigla_uf': 'UF de residência'}, inplace=True)
df_mun_merged

#Mescla dados do FIES com latitude e longitude do município de residência do candidato
df_fies_merged = pd.merge(df_fies, df_mun_merged, on=['Municipio de residência', 'UF de residência'], how='left')
df_fies_merged.describe(include='all')
#Pela quantidade de linhas não nulas de latitude e longitude, observa-se que não se encontrou dados para todos os municípios

#Verifica quais são os municípios onde não se achou lat long
```

```
df_fies_merged[df_fies_merged['latitude'].isnull()][['Municipio de residên-
cia','UF de residência']].value_counts()

#Correção de erro de grafia no nome do município
df_fies.loc[df_fies['Municipio de residência'] == 'MOJI MIRIM','Municipio de
residência'] = 'MOGI MIRIM'
#Correção de erro de UF no município
df_fies.loc[df_fies['Municipio de residência'] == 'ABDON BATISTA', 'UF de re-
sidência'] = 'SC'
#Correção de município que trocou de nome
linhas_filtro_df = (df_fies['Municipio de residência'] == 'PRESIDENTE JUSCELI-
NO') &                    (df_fies['UF de residência'] == 'RN')
df_fies.loc[linhas_filtro_df, 'Municipio de residência'] = 'SERRA CAIADA'
linhas_filtro_df = (df_fies['Municipio de residência'] == 'AUGUSTO SEVERO') &
                   (df_fies['UF de residência'] == 'RN')
df_fies.loc[linhas_filtro_df, 'Municipio de residência'] = 'CAMPO GRANDE'
linhas_filtro_df = (df_fies['Municipio de residência'] == 'CAMPO DE SANTANA')
&                    (df_fies['UF de residência'] == 'PB')
df_fies.loc[linhas_filtro_df, 'Municipio de residência'] = 'TACIMA'
linhas_filtro_df = (df_fies['Municipio de residência'] == 'SANTAREM') &
             (df_fies['UF de residência'] == 'PB')
df_fies.loc[linhas_filtro_df, 'Municipio de residência'] = 'JOCA CLAUDINO'


#Mescla dados do FIES com latitude e longitude do município de residência do
candidato, após ajustes de nomes na base FIES
df_fies_merged = pd.merge(df_fies, df_mun_merged, on=['Municipio de residên-
cia', 'UF de residência'], how='left')
df_fies_merged.describe(include='all')
#Pela quantidade de linhas não nulas de latitude e longitude, observa-se que
ainda não se encontrou dados para todos os municípios

#Verifica quais são os municípios onde ainda não se achou lat long
df_fies_merged[df_fies_merged['latitude'].isnull()][['Municipio de residên-
cia','UF de residência']]

#Verifica como está preenchido o nome na base de municípios
df_mun_merged[df_mun_merged['Municipio de residência'].str.contains('CAMPO
GRANDE')]

#Ajusta o nome do município, retirando o nome antigo
linhas_filtro_df = df_mun_merged['Municipio de residên-
cia'].str.contains('AUGUSTO SEVERO') &                    (df_mun_merged['UF de
residência'] == 'RN')
df_mun_merged.loc[linhas_filtro_df, 'Municipio de residência'] = 'CAMPO GRAN-
DE'


#Verifica como está preenchido o nome na base de municípios
```

```
df_mun_merged[df_mun_merged['Municipio de residência'].str.contains('SERRA
CAIADA')]

#Ajusta o nome do município, retirando o nome antigo
linhas_filtro_df = df_mun_merged['Municipio de residên-
cia'].str.contains('SERRA CAIADA') &                    (df_mun_merged['UF de
residência'] == 'RN')
df_mun_merged.loc[linhas_filtro_df, 'Municipio de residência'] = 'SERRA CAIA-
DA'

#Mescla dados do FIES com latitude e longitude do município de residência do
candidato, após ajustes de nomes na base FIES
df_fies_merged = pd.merge(df_fies, df_mun_merged, on=['Municipio de residên-
cia', 'UF de residência'], how='left')
df_fies_merged.describe(include='all')
#Agora com todos os municípios de residência do candidato com latitude e lon-
gitude

#Renomeia colunas lat long para referenciar município de residencia do candi-
dato do fies
df_fies_merged.rename(columns={'latitude': 'Latitude Residência', 'longitude':
'Longitude Residência'}, inplace=True)

#Renomeia colunas para fazer o merge com os dados de município de oferta do
curso do candidato do fies
df_mun_merged.rename(columns={'Municipio de residência': 'Munícipio do Local
de Oferta', 'UF de residência': 'UF do Local de Oferta'}, inplace=True)
df_mun_merged

#Mescla dados do FIES com latitude e longitude do município de residência do
candidato
df_fies_merged = pd.merge(df_fies_merged, df_mun_merged, on=['Munícipio do
Local de Oferta', 'UF do Local de Oferta'], how='left')
df_fies_merged.describe(include='all')
#Verifica que todos os municípios de oferta do curso do candidato possuem la-
titude e longitude

#Renomeia colunas lat long para referenciar município de oferta do candidato
do fies
df_fies_merged.rename(columns={'latitude': 'Latitude Oferta', 'longitude':
'Longitude Oferta'}, inplace=True)

#Serão utilizadas as coordenadas geográficas do município do candidato e do
município de oferta do curso selecionado.
#Optou-se por eliminar os atributos relativos à UF do município, para que se
privilegie o critério da distância geográfica e não
#o da divisão política. Por exemplo, candidatos que moram em cidades próximas
deveriam ser aproximados, mesmo que as cidades
#sejam de UF distintas (regiões de fronteira).
```

```
del df_fies_merged['Municipio de residência']
del df_fies_merged['Munícipio do Local de Oferta']
del df_fies_merged['UF de residência']
del df_fies_merged['UF do Local de Oferta']

df_fies_merged.info()


#Visualiza a correlação dos atributos via mapa de calor
sns.set()
plt.figure(figsize = (12, 9))
s = sns.heatmap(df_fies_merged.corr(),
                annot = True,
                cmap = 'RdBu',
                vmin = -1,
                vmax = 1)
s.set_yticklabels(s.get_yticklabels(), rotation = 0, fontsize = 12)
s.set_xticklabels(s.get_xticklabels(), rotation = 90, fontsize = 12)
plt.title('Correlation Heatmap')
plt.show()
#Percebe-se uma alta correlação positiva entre latitude e longitude do municí-
pio de residência do estudante e
#do município de oferta do curso, o que faz sentido. O candidato busca um cur-
so na própria cidade ou em cidade próxima à sua.
#Percebe-se também uma alta correlação negativa entre a idade do candidato e o
ano de conclusão do ensino médio.
#Ou seja, quanto mais velho o candidato, mais provável que tenha terminado o
ensino médio mais cedo, o que é esperado.


# # Tratamento dos atributos categóricos
#
# Todos os 7 atributos categóricos são atributos categóricos nominais. Será
feito primeiro a codificação via label encoder para transformação de dados
textuais em numéricos e depois via one hot encoder (apenas para os atributos
categóricos não binários).

#Usa amostra de 10 mil linhas aleatórias para o processamento, para evitar
demora no processamento de todo o dataframe
df_fies_amostra = df_fies_merged.sample(n=10000, random_state=42)

fies_lab_encoded = df_fies_amostra.iloc[:].values
fies_lab_encoded


#Codificação dos dados categóricos via label encoder
label_encoder_Sexo = LabelEncoder()
label_encoder_Etnia = LabelEncoder()
label_encoder_PCD = LabelEncoder()
```

```
label_encoder_TpEscola = LabelEncoder()
label_encoder_CursoSup = LabelEncoder()
label_encoder_Prof = LabelEncoder()
label_encoder_GrupPref = LabelEncoder()
label_encoder_NomeCurso = LabelEncoder()
label_encoder_Turno = LabelEncoder()

fies_lab_encoded[:,0] = la-
bel_encoder_Sexo.fit_transform(fies_lab_encoded[:,0])
fies_lab_encoded[:,1] = la-
bel_encoder_Etnia.fit_transform(fies_lab_encoded[:,1])
fies_lab_encoded[:,2] = label_encoder_PCD.fit_transform(fies_lab_encoded[:,2])
fies_lab_encoded[:,3] = la-
bel_encoder_TpEscola.fit_transform(fies_lab_encoded[:,3])
fies_lab_encoded[:,5] = la-
bel_encoder_CursoSup.fit_transform(fies_lab_encoded[:,5])
fies_lab_encoded[:,6] = la-
bel_encoder_Prof.fit_transform(fies_lab_encoded[:,6])
fies_lab_encoded[:,10] = la-
bel_encoder_NomeCurso.fit_transform(fies_lab_encoded[:,10])
fies_lab_encoded[:,11] = la-
bel_encoder_Turno.fit_transform(fies_lab_encoded[:,11])
fies_lab_encoded.shape

#Codificação dos dados categóricos não binários (Etnia/Cor, Tipo de escola,
Nome do Curso e Turno) via one hot encoder
onehotencoder_fies = ColumnTransformer(transformers=[('OneHot', OneHotEncod-
er(), [1,3,10,11])], remainder='passthrough')
fies_ohe_encoded = onehotencoder_fies.fit_transform(fies_lab_encoded)
fies_ohe_encoded


# # Normalização dos dados

# Escalonamento dos valores via padronização
scaler_fies = StandardScaler(with_mean=False)
fies_scaled = scaler_fies.fit_transform(fies_ohe_encoded)
fies_scaled


# # Clusterização Hierárquica

#Aplica o dendograma para encontrar a quantidade ideal de clusters
dendrograma = dendrogram(linkage(fies_scaled.toarray(), method='ward'),
                         truncate_mode = 'level',
                         p = 5,
                         show_leaf_counts = False,
                         no_labels = True)
plt.title('Dendrograma')
```

```
plt.xlabel('Candidatos')
plt.ylabel('Distância');
#No agrupamento hierárquico a quantidade ideal de clusters é dada pela maior
linha vertical no dendograma que não é cortada
#por nenhuma linha horizontal do diagrama. Neste caso a quantidade de clusters
é 2


#Aplica o algoritmo de agrupamento hierárquico com 2 clusters
hc_fies = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage
= 'ward')
hc_rotulos = hc_fies.fit_predict(fies_scaled.toarray())
np.unique(hc_rotulos, return_counts=True)

#Cria dataframe com cópia dos dados analisados, anexando coluna com a classi-
ficação pelo agrupamento hierárquico
df_fies_clust = df_fies_amostra.copy()
df_fies_clust['Cluster HC'] = hc_rotulos

# ## Análise de resultado do HC
df_fies_clust[df_fies_clust['Cluster HC'] == 0].describe()
df_fies_clust[df_fies_clust['Cluster HC'] == 1].describe()
df_fies_clust[df_fies_clust['Cluster HC'] == 0].describe(include=[object])
df_fies_clust[df_fies_clust['Cluster HC'] == 1].describe(include=[object])

# # Clusterização pelo algoritmo K-Means
#Escolha do número de clusters pelo método do Cotovelo. O valor ideal é dado
pelo ponto de inflexão mais acentuado.
sse = []
k_range = range(1, 11)
for k in k_range:
    print(dt.now())
    kmeans = KMeans(n_clusters=k, init = 'k-means++', random_state = 0)
    kmeans.fit(fies_scaled)
    sse.append(kmeans.inertia_)
plt.style.use("fivethirtyeight")
plt.plot(k_range, sse)
plt.xticks(k_range)
plt.xlabel("Número de Clusters")
plt.ylabel("SSE")
plt.title('Método do Cotovelo para o K-Means')
plt.show()

#Embora o método do cotovelo e o da silhueta sejam técnicas subjetivas, em
ambos observou-se a quantidade ideal de clusters 2
#Aplica o algoritmo KMeans com 2 clusters
kmeans = KMeans(n_clusters=2, init = 'k-means++', random_state = 0)
kmeans_rotulos = kmeans.fit_predict(fies_scaled)
np.unique(kmeans_rotulos, return_counts=True)
```

```
#Anexa coluna com a classificação pelo agrupamento K-means
df_fies_clust['Cluster K-means'] = kmeans_rotulos


# ## Análise de resultado do K-means
df_fies_clust[df_fies_clust['Cluster K-means'] == 0].describe()
df_fies_clust[df_fies_clust['Cluster K-means'] == 1].describe()
df_fies_clust[df_fies_clust['Cluster K-means'] ==
0].describe(include=[object])
df_fies_clust[df_fies_clust['Cluster K-means'] ==
1].describe(include=[object])

# # Clusterização pelo algoritmo PAM (K-medoids)
#Escolha do número de clusters pelo método do Cotovelo. O valor ideal é dado
pelo ponto de inflexão mais acentuado.
sse = []
k_range = range(1, 11)
for k in k_range:
    print(dt.now())
    pam = KMedoids(n_clusters=k)
    pam.fit(fies_scaled)
    sse.append(pam.inertia_)
plt.style.use("fivethirtyeight")
plt.plot(k_range, sse)
plt.xticks(k_range)
plt.xlabel("Número de Clusters")
plt.ylabel("SSE")
plt.title('Método do Cotovelo para o PAM')
plt.show()

#Aplica o algoritmo PAM com 2 clusters
pam = KMedoids(n_clusters=2).fit(fies_scaled)
np.unique(pam.labels_, return_counts=True)

#Anexa coluna com a classificação pelo agrupamento PAM
df_fies_clust['Cluster PAM'] = pam.labels_


# ## Análise de resultado do PAM
df_fies_clust[df_fies_clust['Cluster PAM'] == 0].describe()
df_fies_clust[df_fies_clust['Cluster PAM'] == 1].describe()
df_fies_clust[df_fies_clust['Cluster PAM'] == 0].describe(include=[object])
df_fies_clust[df_fies_clust['Cluster PAM'] == 1].describe(include=[object])


# # Clusterização pelo algoritmo DBScan
#Usa o método do cotovelo para encontrar o valor ideal de eps, no ponto onda
há uma inflexão na curva.
```

```
k = 4
nbrs = NearestNeighbors(n_neighbors=k+1).fit(fies_scaled)
distances, indices = nbrs.kneighbors(fies_scaled)
k_dist = distances[:, -1]

# Ordene as distâncias em ordem crescente
sorted_distances = np.sort(k_dist)

plt.plot(np.arange(len(sorted_distances)), sorted_distances)
plt.ylim(0,9)
plt.xlabel('Índice do ponto')
plt.ylabel('eps')
plt.title('Método do cotovelo para o valor ideal de eps')
plt.show()


# Uma forma de encontrar os valores ideais para os parâmetros do DBSCAN é uti-
lizando a técnica de busca em grade (grid search). Nessa técnica, uma grade de
valores é definida para cada parâmetro a ser ajustado e o algoritmo é executa-
do em cada combinação possível de valores dos parâmetros, sendo avaliado atra-
vés de uma métrica de desempenho pré-definida, neste caso o score da silhueta.
# Definindo a grade de valores para os parâmetros
param_grid = {'eps': np.arange(5, 8, 0.2), 'min_samples': np.arange(2, 10, 1)}

# Criando um modelo DBSCAN sem parâmetros definidos
dbscan = DBSCAN()

# Realizando a busca em grade
best_score_silhouette = -1
best_eps_silhouette = 0
best_min_samples_silhouette = 0
for eps in param_grid['eps']:
    for min_samples in param_grid['min_samples']:
        dbscan.set_params(eps=eps, min_samples=min_samples)
        dbscan.fit(fies_scaled)
        if len(set(dbscan.labels_)) > 1:
            score_silhouette= silhouette_score(fies_scaled, dbscan.labels_)
            if score_silhouette > best_score_silhouette:
                best_score_silhouette = score_silhouette
                best_eps_silhouette = eps
                best_min_samples_silhouette = min_samples
        print('eps:', eps, 'min_samples:', min_samples, 'ruído:',
np.unique(dbscan.labels_, return_counts=True)[1][0],
              'score_silhouette:', score_silhouette)

# Imprimindo os melhores valores para os parâmetros
print("Melhor valor para eps: ", best_eps_silhouette)
print("Melhor valor para min_samples: ", best_min_samples_silhouette)
```

```
dbscan = DBSCAN(eps = best_eps_silhouette,
min_samples=best_min_samples_silhouette)
dbscan.fit(fies_scaled)
np.unique(dbscan.labels_, return_counts=True)

#Anexa coluna com a classificação pelo agrupamento DBScan
df_fies_clust['Cluster DBScan'] = dbscan.labels_


# ## Análise de resultado do DBScan
df_fies_clust[df_fies_clust['Cluster DBScan'] == 2].describe()
df_fies_clust.describe()
df_fies_clust[df_fies_clust['Cluster DBScan'] == 3].describe(include=[object])
df_fies_clust.describe(include=[object])


# # Análise de redução de dimensionalidade via PCA
#Avalia a  redução das dimensões utilizando o Principal Component Analysis
(PCA).
pca = PCA(n_components=fies_scaled.shape[1])
pca.fit(fies_scaled.toarray())
variance = pca.explained_variance_ratio_
var = np.cumsum(np.round(variance, 3)*100)
plt.figure(figsize=(12,6))
plt.ylabel('% Variância Explicada')
plt.xlabel('# de Features')
plt.title('Análise PCA')
plt.xlim(0,fies_scaled.shape[1])
plt.plot(var, marker = 'o')


# # Visualização dos resultados do agrupamento
fig = px.scatter_mapbox(df_fies_clust, lat="Latitude Residência",
lon="Longitude Residência", color="Cluster HC",
                        hover_name="Cluster HC",
                        hover_data=["Sexo","Idade","Nome do curso","Renda men-
sal bruta per capita"], zoom=10)
fig.show()

fig = px.scatter_mapbox(df_fies_clust, lat="Latitude Residência",
lon="Longitude Residência", color="Cluster K-means",
                        hover_name="Cluster K-means",
                        hover_data=["Sexo","Idade","Nome do curso","Renda men-
sal bruta per capita"], zoom=10)
fig.show()


fig = px.scatter_mapbox(df_fies_clust, lat="Latitude Residência",
lon="Longitude Residência", color="Cluster PAM",
```

```
                        hover_name="Cluster PAM",
                        hover_data=["Sexo","Idade","Nome do curso","Renda men-
sal bruta per capita"], zoom=10)
fig.show()
```