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

Mateus Felipe Santos Araújo

PREVISÃO DE NOTAS DE CORTE DO ENEM

Belo Horizonte
2024

Mateus Felipe Santos Araújo

PREVISÃO DE NOTAS DE CORTE DO ENEM

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
2024

SUMÁRIO

1. Introdução

1.1. Contextualização

O Exame Nacional do Ensino Médio (ENEM) é a principal porta de entrada para o ensino superior no Brasil, logo a previsão das notas de corte para os cursos ofertados pelas universidades brasileiras é vital para os estudantes e instituições. Isso ajudará os estudantes a obter orientação adicional e a tomar decisões bem informadas na escolha de cursos e universidades.

Dessa forma, os estudantes aumentam suas chances de aprovação, ao mesmo tempo que reduzem o estresse e a incerteza. Para as universidades, a previsão oferece uma oportunidade das instituições se adaptarem às tendências observadas e ajustar suas políticas. Para a sociedade como um todo, a implementação de modelos preditivos no campo educacional é um avanço em direção a um sistema mais transparente e eficaz.

Em resumo, um modelo preditivo para as notas do ENEM é relevante e significativo, considerando que beneficia os alunos, universidades e a sociedade e promove uma educação mais justa.

1.2. O problema proposto

O objetivo da previsão das notas do ENEM utilizando técnicas de machine learning é automatizar e aprimorar o processo de previsão das notas de corte. Em vez de depender exclusivamente de análises manuais ou estimativas baseadas em dados históricos, o machine learning utiliza algoritmos e técnicas avançadas para analisar os dados e aprender padrões. Isso permite prever as notas de forma mais eficiente e precisa, beneficiando tanto os estudantes, que podem fazer escolhas mais fundamentadas, quanto as universidades, que podem planejar melhor a alocação de vagas e recursos.

As informações utilizadas neste projeto foram coletadas a partir dos dados públicos disponíveis nos sites do MEC (Ministério da Educação) e do IBGE (Instituto Brasileiro de Geografia e Estatística). A base de dados escolhida para o projeto é

composta pelas notas de corte dos cursos de graduação das universidades cadastradas no SISU (Sistema de Seleção Unificada) de 2023. Esse material contém informações de 44 mil cursos cadastrados no ENEM daquele ano.

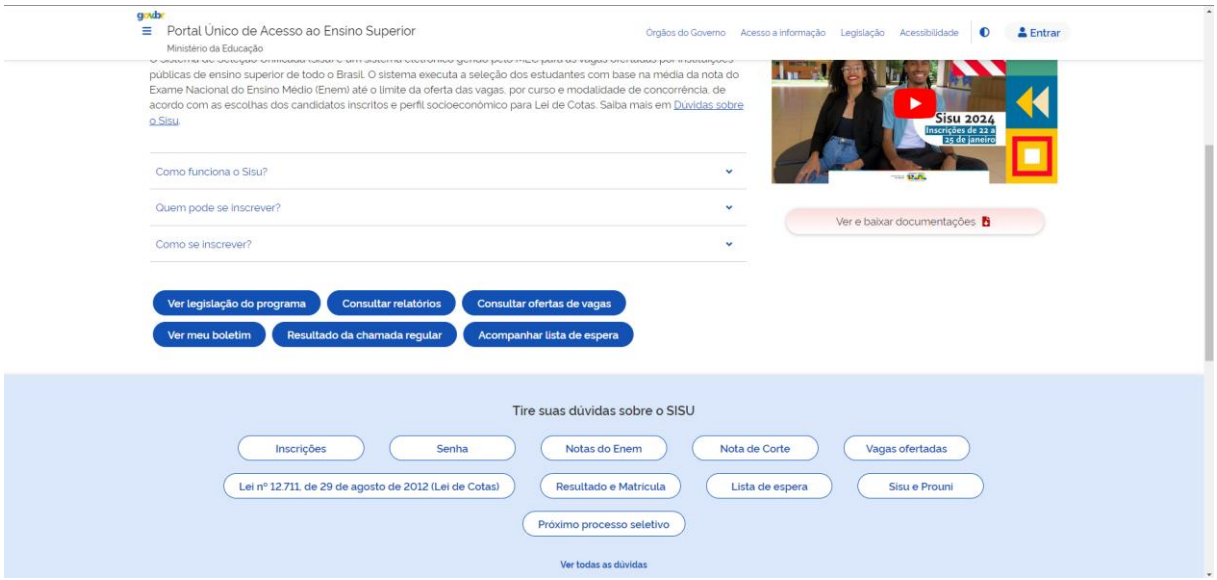
Nesse trabalho será analisado o *dataset* citado anteriormente, juntamente com informações geográficas e institucionais, com o objetivo de criar um modelo de *machine learning* capaz de prever a nota de corte de um determinado curso com base nas informações disponíveis no conjunto de dados.

2. Coleta de Dados

2.1. Base de Dados Principal

O conjunto de dados escolhido para elaboração deste trabalho foi obtido a partir dos dados públicos dos cursos cadastrados no SISU de 2023, baixados em 29/02/2024, por meio do link https://acessounico.mec.gov.br/sisu. No site é necessário rolar a página para baixo e clicar em "**Consultar relatórios**".

Figura 1: Print do site do MEC



Na seção direcionada, foi selecionada a base "**Sisu 2023/1_Inscrições e notas de corte**". Com isso foi possível obter informações detalhadas sobre os cursos e faculdades participantes do ENEM de 2023, incluindo as notas de corte. A base de dados é composta pelos seguintes campos:

Tabela 1: Estrutura da Tabela dos dados do SISU de 2023

Nome da coluna/campo	Descrição	Tipo
NU_ANO	Ano do processo seletivo	Inteiro
NU_EDICAO	Número da edição do processo seletivo no ano de referência	Inteiro
CO_IES	Código da instituição de ensino superior conforme informações do cadastro e-MEC	String
NO_IES	Nome da instituição de ensino superior conforme informações do cadastro e-MEC	String
SG_IES	Sigla da instituição de ensino superior conforme informações do cadastro e-MEC	String
DS_ORGANIZACAO_ACADEMICA	Descrição da organização acadêmica da instituição de ensino superior conforme informações do cadastro e-MEC	String
DS_CATEGORIA_ADM	Descrição da categoria administrativa da instituição de ensino superior conforme informações do cadastro e-MEC	String
NO_CAMPUS	Nome do campus da	String

	instituição de ensino superior conforme informações do cadastro e-MEC	
NO_MUNICIPIO_CAMPUS	Nome do município do campus da instituição de ensino superior conforme informações do cadastro e-MEC	String
SG_UF_CAMPUS	Sigla da unidade da federação (UF) na qual está localizada o campus da instituição de ensino superior conforme informações do cadastro e-MEC	String
DS_REGIAO_CAMPUS	Descrição da região na qual está localizada o campus da instituição de ensino superior conforme informações do cadastro e-MEC	String
CO_IES_CURSO	Código do curso da instituição de ensino superior conforme informações do cadastro e-MEC	String
NO_CURSO	Nome da instituição de ensino superior conforme informações do cadastro e-MEC	String
DS_GRAU	Grau do curso da instituição de ensino	String

	superior conforme informações do cadastro e-MEC	
DS_TURNO	Turno do curso da instituição de ensino superior conforme informações do cadastro e-MEC	String
TP_MODALIDADE	Tipo da modalidade da oferta do curso no processo seletivo	String
DS_MOD_CONCORRENCIA	Descrição do tipo da modalidade de concorrência ofertada para o curso no processo seletivo	String
NU_PERCENTUAL_BONUS	Percentual de bônus definido para as ações afirmativas próprias das IES	Decimal
QT_VAGAS_CONCORRENCIA	Quantidade de vagas ofertadas naquela modalidade	Inteiro
NU_NOTACORTE	Nota de corte da modalidade/curso conforme o resultado da chamada regular	Decimal
QT_INSCRICAO	Quantidade de inscrições para a modalidade	Inteiro

Nota: A nota de corte é sempre igual a nota do último candidato classificado na última vaga para a modalidade/curso escolhida. Na modalidade de bônus a nota de corte é a mesma da modalidade ampla concorrência do mesmo curso porque as

pessoas que se inscreveram na modalidade bônus concorrem pelas vagas ofertadas para ampla concorrência do mesmo curso/grau/turno/campus/IES. Alguns cursos podem não apresentar nota de corte pois não tiveram número de inscritos pelo menos igual ao número de vagas ou porque tiveram poucos inscritos e alguns desses inscritos tiveram seleção na primeira opção de curso e, portanto, não têm sua nota considerada para sua segunda opção de inscrição.

Os dados do SISU constituem a base principal do projeto, contendo 44 mil cursos cadastrados no sistema de 2023, além de diversas informações institucionais, geográficas e relacionadas aos resultados do ENEM após a aplicação e seleção dos candidatos.

2.2. Bases de Dados Complementares

Além dos dados principais obtidos do SISU, este trabalho também fez a utilização de bases complementares. A primeira foi retirada do site do e-MEC: Sistema de Regulação do Ensino Superior (https://emec.mec.gov.br/emec/nova#avancada). A consulta aos dados foi feita no dia 06/03/2024. O e-MEC é o sistema oficial de cadastro e regulação dos cursos e instituições de ensino superior no Brasil, conforme regulamentado pela Portaria Normativa nº 21, de 21/12/2017.

O e-MEC fornece informações detalhadas sobre as instituições de ensino superior, que dependem da validade dos respectivos atos autorizativos e do protocolo tempestivo dos processos regulatórios de manutenção de autorização para o funcionamento das instituições e a oferta de cursos. Os dados registrados no e-MEC são de responsabilidade das próprias instituições de ensino, sejam elas federais, estaduais, ou municipais, garantindo que as informações estejam de acordo com os atos autorizativos emitidos pelo Poder Público ou órgão competente. Os atributos desejados desta base são os índices educacionais que o MEC utiliza para avaliar as universidades.

Para este estudo, a base do e-MEC foi utilizada para complementar as informações institucionais com dados adicionais sobre as universidades. A base do

e-MEC consiste em 365 registros de universidades cadastradas, cada uma descrita por 28 atributos diferentes. No entanto, para fins de análise, apenas 6 desses atributos foram selecionados por sua relevância para análise:

Tabela 2: Estrutura da Tabela dos cursos cadastrados no e-MEC

Nome da coluna/campo	Descrição	Tipo
Código IES	Código da instituição de ensino superior conforme o cadastro no e-MEC	String
Instituição(IES)	Nome da instituição de ensino superior conforme o cadastro no e-MEC	String
Município	Nome do município onde está localizada a instituição de ensino superior	String
UF	Sigla da unidade da federação (UF) na qual está localizada a instituição de ensino superior	String
CI	Conceito Institucional	Inteiro
IGC	Índice Geral de Cursos	Inteiro

Nota: Conceito Institucional é atribuído pelo MEC após visitas presenciais para avaliação das instalações físicas, políticas de gestão e plano de desenvolvimento institucional. O Índice Geral de Cursos é a média das notas (CPC) de todos os cursos de graduação e pós-graduação oferecidos pela instituição de ensino, divulgado anualmente.

Essas informações complementares do e-MEC são essenciais para enriquecer a análise, proporcionando um contexto mais abrangente sobre as instituições de ensino superior, suas localizações geográficas, e seus desempenhos institucionais.

Este projeto também fez uso de informações geográficas e demográficas provenientes do Instituto Brasileiro de Geografia e Estatística (IBGE). O objetivo foi complementar as informações institucionais e acadêmicas com dados socioeconômicos que permitam uma análise mais abrangente das universidades e dos cursos ofertados.

A base do IBGE foi utilizada para enriquecer os dados coletados do SISU, adicionando características populacionais dos municípios onde as instituições de ensino superior estão localizadas. As informações sobre a população municipal foram extraídas do site do IBGE no link https://sidra.ibge.gov.br/pesquisa/censo-demografico/demografico-2022/primeiros-resultados-populacao-e-domicilios no dia 07/03/2024. Este conjunto de dados contém as populações dos 5.570 municípios do Brasil, conforme o Censo de 2022.

Para este trabalho, foram selecionadas três colunas principais da base de dados do IBGE:

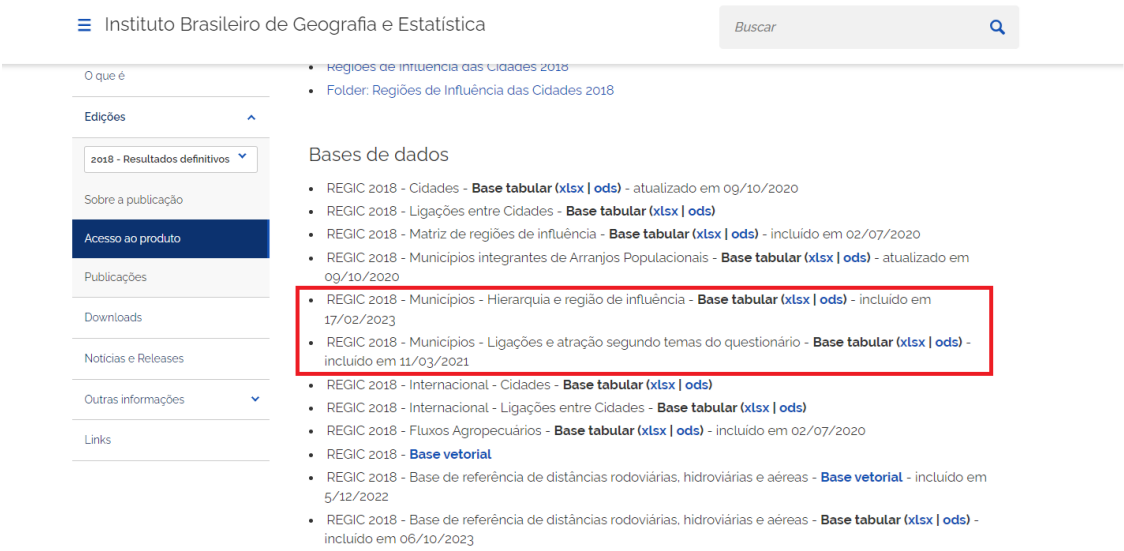
Tabela 3: Estrutura da Tabela da População dos Municípios pelo Censo de 2022

Nome da coluna/campo	Descrição	Tipo
Cód.	Código do IBGE do Município	String
Município	Nome do Município e a sigla da UF	String
Ano	População do município recenseada no Censo de 2022	Inteiro

A integração desses dados ao projeto permite explorar como fatores demográficos e geográficos podem influenciar na escolha dos estudantes às instituições de ensino superior. Por exemplo, a análise da população municipal pode ajudar a identificar padrões de demanda por educação superior em diferentes regiões, enquanto o uso do código do município do IBGE facilita a incorporação de outras características urbanas e socioeconômicas, como a Hierarquia Urbana e os índices de Atração dos municípios. Essas características foram captadas no site do IBGE (https://www.ibge.gov.br/geociencias/cartas-e-mapas/redes-

geograficas/15798-regioes-de-influencia-das-cidades.html), acessadas no dia 07/03/2024. Essas bases fornecem informações detalhadas sobre a hierarquia urbana e os índices de atração dos municípios brasileiros, que irá auxiliar na compreensão do contexto socioeconômico dos locais onde estão localizadas as instituições de ensino superior.

Figura 2: Print das bases retiradas do site do IBGE



A **Hierarquia Urbana** indica a centralidade de uma cidade com base na atração que exerce sobre as populações de outros centros urbanos para o acesso a bens e serviços, além do nível de articulação territorial que a cidade possui devido à sua inserção em atividades de gestão pública e empresarial. Esta base categoriza os municípios brasileiros em cinco níveis hierárquicos, com subdivisões específicas: Metrópoles (1A, 1B e 1C), Capitais Regionais (2A, 2B e 2C), Centros Sub-Regionais (3A e 3B), Centros de Zona (4A e 4B) e Centros Locais (5). A última categorização foi realizada em 2018.

As colunas principais da base de Hierarquia urbana são:

Tabela 4: Estrutura da Tabela de Hierarquia Urbana de Municípios

Nome da coluna/campo	Descrição	Tipo

siguf	Sigla da Unidade da federação	String
codmun	Código do município	String
Hierarquia 2	Descrição da Hierarquia	String
Hierarquia 3	Código da Hierarquia	String
Hierarquia - grupo	Classificação da hierarquia urbana por grupos, onde 1 representa Metrópoles, 2 representa Capitais Regionais, 3 representa Centros Sub-Regionais, 4 representa Centros de Zona, e 5 representa Centros Locais	Strong

A segunda base, referente ao **Índice de Atração**, avalia a atração dos municípios como destino para residentes de outras localidades. Foram extraídos dois índices principais desta base: o índice geral de atração e o índice de atração específico para o ensino superior. Esses índices são fundamentais para entender como os municípios atraem pessoas para suprir bens e serviços, incluindo a busca por educação superior. A última medição do atributo foi realizada em 2018.

As colunas principais da base de Atratividade são:

Tabela 5: Estrutura da Tabela dos Índices de Atração dos Municípios

Nome da coluna/campo	Descrição	Tipo
CODMUN	Código de município	String
IA	Índice de atração geral como destino para moradores de outros municípios para consumo de bens e serviços	String
IA_Q5	Índice de atração como destino para moradores	String

	de outros municípios para ensino superior	

Esses atributos adicionais são fundamentais para compreender o contexto em que as universidades operam, bem como para prever as notas de corte com maior precisão. A análise integrada desses dados possibilita uma visão mais ampla e detalhada das dinâmicas educacionais no Brasil, considerando não apenas os dados acadêmicos, mas também o impacto das condições demográficas e geográficas na educação superior.

Com uma base de dados robusta, o projeto busca construir um modelo preditivo que capture as nuances e complexidades do sistema educacional brasileiro, fornecendo insights valiosos para estudantes, universidades e formuladores de políticas públicas com a previsão das notas de corte dos cursos.

3. Processamento/Tratamento de Dados

3.1. Base de Dados Principal

Para realizar o projeto foi utilizado o software *Python*, dentro do ambiente foram importadas algumas bibliotecas básicas para primeira parte do estudo, que consiste na coleta, processamento e tratamento dos dados.

Figura 3: Importação de Bibliotecas do Python

```
# Importando bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from unidecode import unidecode
```

Primeiramente faz-se a importação da base principal do projeto, das notas do SISU. A base consiste em 44326 linhas e 21 colunas, com as notas de todos os cursos cadastrados no sistema de 2023 e suas características.

Figura 4: Importação da base do SISU

```
sisu = pd.read_excel('DADOS/portal_sisu_sisu 2023_1_inscricoes_notas_de_corte.xlsx', sheet_name='inscricao_2023_1', index=False)
sisu.shape
```

```
(44326, 21)
```

Com o conhecimento de alguns processos do sistema de seleção única (SISU), torna-se necessária a análises de determinados atributos. Quando um aluno se cadastra no SISU, ele é alocado em grupos específicos, cujas características influenciam significamente o grupo no qual o estudante irá “disputar” a tão sonhada vaga na universidade. Estes grupos refletem grandes diferenças nas notas de corte com fatores específicos de cada população que influenciam na nota de corte, portanto, foram realizados alguns ajustes nos dados, com o objetivo de tornar o modelo focado em uma parte específica dos alunos.

A primeira informação a ser considerada é a **modalidade de concorrência** em que o aluno se cadastra. Foi excluído da análise as modalidades de concorrência específicas, como cotas e vagas especiais, mantendo apenas a modalidade de ampla concorrência. Essa escolha se justifica pela diversidade de critérios utilizados nas modalidades de cotas, como renda, raça, escola pública, entre outros, que introduzem uma diferença significativa nas notas de corte sem influência de critérios específicos de reserva de vagas.

Figura 5: Modalidades de Concorrência presentes na base

```
sisu.TP_MOD_CONCORRENCIA.value_counts()
```

```
L    31222
A     6402
V     5554
B     1148
Name: TP_MOD_CONCORRENCIA, dtype: int64
```

```
sisu.DS_MOD_CONCORRENCIA.value_counts()
```

```
Ampla concorrência
6402
Candidatos com renda familiar bruta per capita igual ou inferior a 1,5 salário mínimo que tenham cursado integralmente o ensino
médio em escolas públicas (Lei nº 12.711/2012).
4845
Candidatos que, independentemente da renda (art. 14, II, Portaria Normativa nº 18/2012), tenham cursado integralmente o ensino
médio em escolas públicas (Lei nº 12.711/2012).
4738
Candidatos autodeclarados pretos, pardos ou indígenas, com renda familiar bruta per capita igual ou inferior a 1,5 salário míni
mo e que tenham cursado integralmente o ensino médio em escolas públicas (Lei nº 12.711/2012).
4686
Candidatos autodeclarados pretos, pardos ou indígenas que, independentemente da renda (art. 14, II, Portaria Normativa nº 18/20
12), tenham cursado integralmente o ensino médio em escolas públicas (Lei nº 12.711/2012).
4639

...
Candidato (s) deficientes auditivos especificamente no curso de Letras-Libras.
1
Candidato (s) surdos
1
concorrentes às vagas de ampla concorrência para o Curso de Medicina - Natal que tiverem cursado todo o ensino médio em escolas
regulares do Estado do Rio Grande do Norte, excluídos aqueles que concluíram o ensino médio por meio de exames supletivos, Exam
e Nacional para Certificação de Competências de Jovens e Adultos - ENCCEJA e equivalentes.       1
concorrentes às vagas de ampla concorrência para o Curso de Medicina - Campus Imperatriz que concluíram todo o Ensino Médio (1
º, 2º e 3º ano) em escolas regulares e presenciais do entorno de um raio de 150 km do município de Imperatriz, conforme Art. 4º
da Res. CONSEPE nº 2.648-2022.                                                                                          1
surdos que optarem pelas vagas destinadas à ampla concorrência terão bonificação de 20% na pontuação final do Exame Nacional do
Ensino Médio (ENEM).
1
Name: DS_MOD_CONCORRENCIA, Length: 199, dtype: int64
```

Após o corte da modalidade a base de dados conta com 6402 linhas e as mesmas 21 colunas.

Figura 6: Corte de Modalidade de Concorrência

```
sisu_a = sisu[sisu.TP_MOD_CONCORRENCIA=="A"].copy()
```

```
sisu_a.shape
```

```
(6402, 21)
```

A segunda informação a ser avaliada é o grau do curso da instituição. Optou-se por excluir os cursos tecnológicos da análise e manter apenas os cursos de bacharelado, licenciatura e ABI (Área Básica de Ingresso). Essa decisão foi tomada devido às diferenças significativas na estrutura, duração e foco dos cursos tecnológicos em comparação aos demais. Ao manter apenas os cursos de bacharelado, licenciatura e ABI é possível garantir maior homogeneidade na análise,

resultando em um modelo mais representativo para o público-alvo principal. A base diminui o número de linhas para 5968 registros.

Figura 7: Corte do Grau do Curso

```
sisu_a = sisu_a[sisu_a.DS_GRAU!='Tecnológico']
sisu_a.shape
```

```
(5968, 21)
```

Também foi analisada a quantidade de vagas ofertadas na modalidade do curso, observando a tabela descritiva da variável foi visualizado que alguns cursos não obtiveram oferta de vaga. Essa informação não faz sentido para análise, pois a nota de corte, foco do projeto, é influenciada não tendo oferta na modalidade de ampla concorrência. Com este primeiro corte a base se encontra com 5961 linhas.

Figura 8: Primeiro corte da quantidade de Vagas ofertadas

```
sisu_a.QT_VAGAS_CONCORRENCIA.describe()
```

```
count    5968.000000
mean       17.303619
std        13.351947
min         0.000000
25%        10.000000
50%        16.000000
75%        22.000000
max       307.000000
Name: QT_VAGAS_CONCORRENCIA, dtype: float64
```

```
sisu_a = sisu_a[sisu_a.QT_VAGAS_CONCORRENCIA!=0]
sisu_a.shape
```

```
(5961, 21)
```

A nota de corte é a variável predita do projeto, ao fazer a descrição dela, é possível observar que a informação *Count* (frequência), na primeira linha da tabela, informa que possui menos registros que a quantidade de linhas da base (5961). Foi decidido por retirar os valores nulos de nota de corte da base, devido a ser uma informação difícil de ser imputada e não oferecer qualidade ao projeto. Após o corte a base possui 5817 registros de cursos.

Figura 9: Corte na Nota de Corte

```
sisu_a.NU_NOTACORTE.describe()
```

```
count    5817.000000
mean      644.547430
std        71.558542
min       323.180000
25%       595.660000
50%       645.120000
75%       693.870000
max       923.900000
Name: NU_NOTACORTE, dtype: float64
```

```
sisu_a = sisu_a[pd.isna(sisu_a.NU_NOTACORTE)!=True]
sisu_a.shape
```

```
(5817, 21)
```

O segundo corte da variável foi realizado devido ao observar valores muito extremos no Boxplot e na Figura **8**. Ao calcular o quantil 99% e 97%, observa-se que os valores estão próximos ainda da distribuição dos dados. Com isso, é decidido fazer a exclusão dos registros com quantidades de vagas ofertadas maior que 51, pois pode ser concluindo que 99% dos cursos possuem menos que 52 vagas ofertadas, não perdendo muita informação e retirando a influência de valores extremos no modelo.

Figura 10: Boxplot da Quantidade de Vagas ofertadas

```
# Criar o boxplot
plt.boxplot(sisu_a.QT_VAGAS_CONCORRENCIA)
# Adicionar título e rótulos dos eixos
plt.title('Boxplot de Quantidade de Vagas ofertadas')
plt.xlabel('')
plt.ylabel('Número de Vagas')

# Exibir o gráfico
plt.show()
```

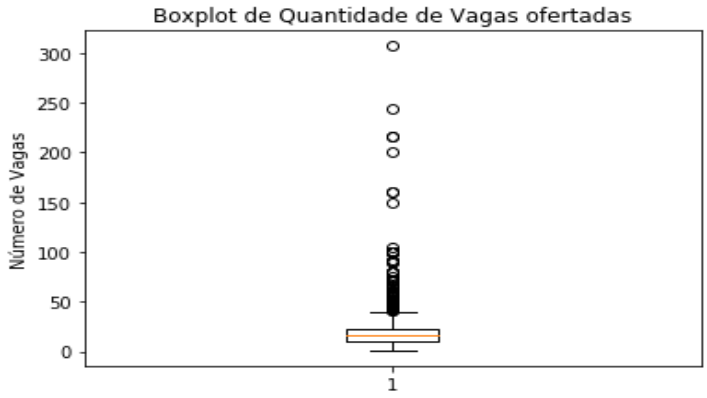


Figura 11: Quantis e Segundo Corte da Quantidade de Vagas ofertadas

```
sisu_a.QT_VAGAS_CONCORRENCIA.quantile([.975, .99])
```

```
0.975    49.6
0.990    51.0
Name: QT_VAGAS_CONCORRENCIA, dtype: float64
```

```
sisu_a = sisu_a[sisu_a.QT_VAGAS_CONCORRENCIA<=51]
sisu_a.shape
```

```
(5759, 21)
```

Por último, foi excluído da análise os cursos na modalidade de Ensino a Distância (EaD), mantendo apenas os cursos presenciais. Essa decisão foi tomada pelos mesmos fatores dos cursos tecnólogos, como diferenças estruturais, perfis dos candidatos, dinâmica de ensino e ainda diferenças geográficas dos alunos. Ao focar apenas nos cursos presenciais, a análise se torna mais consistente e homogênea, garantindo que o modelo seja voltado para o contexto das instituições com uma oferta tradicional de ensino. No final dos cortes, a base principal chega a 5711 registros e 21 atributos.

Figura 12: Corte no Turno dos cursos

```
sisu_a.DS_TURNO.value_counts()
```

```
Integral      2713
Noturno       1895
Matutino       664
Vespertino     439
EaD             48
Name: DS_TURNO, dtype: int64
```

```
sisu_a = sisu_a[sisu_a.DS_TURNO!='EaD']
sisu_a.shape
```

```
(5711, 21)
```

3.2. Bases de Dados Complementares

A primeira base adicionada é do site do E-MEC, a base possui 28 colunas, muitas dessas que não possuem informação relevante para a análise. A intenção de adicionar esta base ao *dataset* principal é obter alguns índices institucionais. Na Figura 14, foi feita a seleção de colunas, mantendo apenas as características importantes. A base do site contempla 365 instituições de ensino superior (IES) e 6 colunas.

Figura 13: Importação da base do site do E-MEC

```
notas_mec = pd.read_excel("DADOS/relatorio_consulta_publica_avancada_ies_06_03_2024_19_52_28.xlsx",skiprows=5,index=False)
```

```
notas_mec.head()
```

	Código Mantenedora	Razão Social	CNPJ	Natureza Jurídica	Código IES	Instituição(IES)	Sigla	Telefone	Sitio	
0	18344	COMANDO DA AERONAUTICA	00.394.429/0111-45	Pessoa Jurídica de Direito Público - Federal	26777	ACADEMIA DA FORÇA AÉREA (AFA)	AFA	NaN	https://www2.fab.mil.br/afa/	
1	18177	CORPO DE BOMBEIROS MILITAR DE MINAS GERAIS	03.389.126/0001-98	Pessoa Jurídica de Direito Público - Estadual	26238	ACADEMIA DE BOMBEIROS MILITAR (ABM)	ABM	3133119179	NaN	abm.ste
2	17963	POLICIA MILITAR DO ESTADO DE MINAS GERAIS	16.695.025/0001-97	Pessoa Jurídica de Direito Público - Estadual	25613	ACADEMIA DE POLÍCIA MILITAR DE MINAS GERAIS (APM)	APM	(31) 2123-9484	https://www.policiamilitar.mg.gov.br/portal-pm...	divisaodeensino.
3	17403	ESPIRITO SANTO SECRETARIA DE EST DE SEGURANCA ...	27.476.373/0001-90	Pessoa Jurídica de Direito Público - Estadual	24465	ACADEMIA DE POLÍCIA MILITAR DO ESPÍRITO SANTO ...	APM/ES	3636-2950	pm.es.gov.br	
4	15797	SANTA CATARINA TRIBUNAL DE JUSTICA	83.845.701/0001-59	Pessoa Jurídica de Direito Público - Estadual	17969	ACADEMIA JUDICIAL DO TRIBUNAL DE JUSTIÇA DE SA...	AJ	48-32872800	http://acadjud.tjsc.jus.br	

5 rows × 28 columns

Figura 14: Seleção de colunas e descrição da base do site E-MEC

```
notas_mec = notas_mec[['Código IES', 'Instituição(IES)', 'Município', 'UF', 'CI', 'IGC']]
```

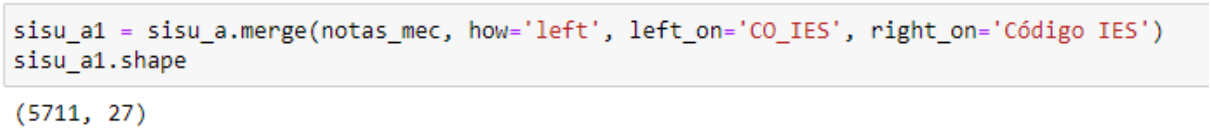
```
notas_mec
```

	Código IES	Instituição(IES)	Município	UF	CI	IGC
0	26777	ACADEMIA DA FORÇA AÉREA (AFA)	NaN	NaN	-	-
1	26238	ACADEMIA DE BOMBEIROS MILITAR (ABM)	NaN	NaN	-	-
2	25613	ACADEMIA DE POLÍCIA MILITAR DE MINAS GERAIS (APM)	Belo Horizonte	MG	-	-
3	24465	ACADEMIA DE POLÍCIA MILITAR DO ESPÍRITO SANTO ...	NaN	NaN	-	-
4	17969	ACADEMIA JUDICIAL DO TRIBUNAL DE JUSTIÇA DE SA...	Florianópolis	SC	5	-
...	...	...	...	...	...	...
360	605	UNIVERSIDADE MUNICIPAL DE SÃO CAETANO DO SUL (...	São Caetano do Sul	SP	-	3
361	76	UNIVERSIDADE REGIONAL DE BLUMENAU (FURB)	Blumenau	SC	-	4
362	746	UNIVERSIDADE REGIONAL DO CARIRI (URCA)	Crato	CE	-	3
363	588	UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UT...	Curitiba	PR	4	4
364	3835	UNIVERSIDADE VIRTUAL DO ESTADO DO MARANHÃO (UN...	São Luís	MA	-	-

365 rows × 6 columns

Com essas informações, é possível realizar a operação de *Join* entre os conjuntos de dados. Utilizando o **código da IES** foi realizado a operação e a base encontrou-se com 5711 registros e 27 colunas. Conforme pode ser visto na Figura 16, apenas 2 linhas não conseguiram correspondência. Essas observações pertencem à faculdade ESCS de Brasília-DF, optou-se por retirar as duas informações da análise, uma vez que a instituição não conta com cadastro ativo no sistema. Por fim, a base ficou com 5709 linhas e 27 atributos.

Figura 15: Operação *Join* entre a base principal e a base do E-MEC



```
sisu_a1 = sisu_a.merge(notas_mec, how='left', left_on='CO_IES', right_on='Código IES')
sisu_a1.shape
```

```
(5711, 27)
```

Figura 16: Corte da faculdade ESCS da base

```
sum(pd.isna(sisu_a1['Município']))
```

```
2
```

```
sisu_a1[pd.isna(sisu_a1['Município'])]
```

	NU_ANO	NU_EDICAO	CO_IES	NO_IES	SG_IES	DS_ORGANIZACAO_ACADEMICA	DS_CATEGORIA_ADM	NO_CAMPUS	NO_MUNICIPIO_CAMPUS	SG_
1306	2023	1	3223	ESCOLA SUPERIOR DE CIÊNCIAS DA SAÚDE	ESCS	Faculdade	Pública Estadual	Brasília-Asa Norte	Brasília	
1307	2023	1	3223	ESCOLA SUPERIOR DE CIÊNCIAS DA SAÚDE	ESCS	Faculdade	Pública Estadual	Curso Graduação de Enfermagem	Brasília	

2 rows × 27 columns

```
sisu_a1 = sisu_a1[sisu_a1.SG_IES!='ESCS']
sisu_a1.shape
```

```
(5709, 27)
```

Com esta operação realizada, conseguiu-se a informação do **município** das instituições. Com isso, serão adicionadas as características demográficas e geográficas ao conjunto final de registros.

Primeiramente foi adicionado os dados do Censo de 2022 do IBGE. A importação da base foi feita levando em consideração algumas informações a mais que vêm com o Excel retirado do site do instituto. Também é decidido por renomear as colunas para facilitar o uso no *script*.

Figura 17: Importação da base do Censo de 2022 do IBGE

```
pop_censo = pd.read_excel("DADOS/pop_censo_2022.xlsx",skiprows=2,index=False)\
            .drop([0,5571],axis=0)\
            .drop('Nível',axis=1)\
            .rename(columns={"Cód.":"COD","Município":"MUNIC","Ano":"POP"})
```

No atributo do município a informação da UF também é encontrada, devido a este motivo é utilizado uma expressão regular para separar o estado do município e salvá-lo em outra coluna.

Figura 18: Tratamento do atributo município e UF

```
pop_censo.MUNIC.head()
```

```
1    Alta Floresta D'Oeste (RO)
2                Ariquemes (RO)
3                   Cabixi (RO)
4                   Cacoal (RO)
5                Cerejeiras (RO)
Name: MUNIC, dtype: object
```

```
# Dividir a coluna "MUNIC" em duas colunas: "MUNIC" e "UF"
pop_censo[['MUNIC', 'UF']] = pop_censo['MUNIC'].str.extract(r'(.+) \((.+)\)')
pop_censo[['MUNIC', 'UF']].head()
```

	MUNIC	UF
1	Alta Floresta D'Oeste	RO
2	Ariquemes	RO
3	Cabixi	RO
4	Cacoal	RO
5	Cerejeiras	RO

Como os códigos do IBGE não estão presentes na base principal, optou-se por utilizar o nome dos municípios e suas respectivas unidades federativas para realizar a operação de *Join* na base de dados principal. Com a função *replace* foi retirado os caracteres especiais dos nomes e depois aplicado a função *Upper* para manter a *string* em maiúsculo. Após as alterações nos dois *datasets* foi feito a operação, resultando em um conjunto com os mesmos 5709 cursos, porém com 30 atributos.

Figura 19: Tratamento do atributo município e operação de *Join* entre as bases

```
# Remover acentos e caracteres especiais dos nomes dos municípios
pop_censo['MUNIC'] = pop_censo['MUNIC'].str.replace(r'[^a-zA-Z\s]', '', regex=True).apply(lambda x: x.upper())

# Alterando Código do Município para STRING
pop_censo["COD"] = pop_censo.COD.astype(int).astype(str)
```

```
sisu_a1["MUNIC"] = sisu_a1['Município'].str.replace(r'[^a-zA-Z\s]', '', regex=True).apply(lambda x: x.upper())
```

```
sisu_a2 = sisu_a1.merge(pop_censo, on=["MUNIC","UF"],how="left")
sisu_a2.shape
```

```
(5709, 30)
```

Neste momento é possível encontrar na base o código do IBGE do município, logo ele é utilizado para extrair as informações de Hierarquia Urbana e Atração dos outros dois conjuntos de dados do IBGE. Nas importações das duas bases são selecionados apenas os atributos relevantes e posteriormente a operação de *Join* é realizada. Ao final da construção da base do projeto, foi então capturado 5709 registros de notas de corte da modalidade de ampla concorrência de cursos presenciais de licenciatura e bacharelado cadastrados do sistema do MEC e 36 atributos são contidos neste *dataset*.

Figura 20: Operação de *Join* entre a base principal e as bases complementares de Hierarquia Urbana e Atração dos Municípios

2.1.2 Hierarquia Urbana

```
regic = pd.read_excel("DADOS/REGIC2018_Municipios_Hierarquia_e_regiao.xlsx",skiprows=0,index=False)\
        [["siguf","codmun","Hierarquia 2","Hierarquia 3","Hierarquia - grupo"]]
regic.columns = ["UF","COD","HIERARQUIA_URBANA","COD_HIERARQUIA_URBANA","GRUPO_HIERARQUIA_URBANA"]
```

```
regic["COD"] = regic.COD.astype(str)
```

```
sisu_a3 = sisu_a2.merge(regic, on=["COD"],how="left")
sisu_a3.shape
```

```
(5709, 34)
```

2.1.3 Atração

```
atracao = pd.read_excel("DADOS/REGIC2018_Quest_Atracao_Municipios.xlsx",skiprows=0,index=False)\
        [["CODMUN","IA","IA_Q5"]]\
        .rename(columns={"CODMUN":"COD"})
```

```
atracao["COD"] = atracao.COD.astype(str)
```

```
sisu_a4 = sisu_a3.merge(atracao, on=["COD"],how="left")
sisu_a4.shape
```

```
(5709, 36)
```

3.3. Tratamento de Dados

Antes de iniciar qualquer tratamento nos dados é necessário conferir se a base está atendendo o objetivo da análise e primeiras explorações nas variáveis devem ser realizadas para definir os tratamentos. A base de dados é importada novamente, e as colunas relevantes são selecionadas. O *dataset* que irá ser utilizado no projeto consta com as 5709 linhas e 19 colunas.

Figura 21: Importação da base e seleção de variáveis

```
sisu = pd.read_csv('sisu_trabalhado.csv')
```

```
sisu.columns
```

```
Index(['NU_ANO', 'NU_EDICAO', 'CO_IES', 'NO_IES', 'SG_IES',
       'DS_ORGANIZACAO_ACADEMICA', 'DS_CATEGORIA_ADM', 'NO_CAMPUS',
       'NO_MUNICIPIO_CAMPUS', 'SG_UF_CAMPUS', 'DS_REGIAO_CAMPUS',
       'CO_IES_CURSO', 'NO_CURSO', 'DS_GRAU', 'DS_TURNO',
       'TP_MOD_CONCORRENCIA', 'DS_MOD_CONCORRENCIA', 'NU_PERCENTUAL_BONUS',
       'QT_VAGAS_CONCORRENCIA', 'NU_NOTACORTE', 'QT_INSCRICAO', 'Código IES',
       'Instituição(IES)', 'Município', 'UF_x', 'CI', 'IGC', 'MUNIC', 'COD',
       'POP', 'UF_y', 'HIERARQUIA_URBANA', 'COD_HIERARQUIA_URBANA',
       'GRUPO_HIERARQUIA_URBANA', 'IA', 'IA_Q5'],
      dtype='object')
```

```
sisu = sisu[["SG_IES","DS_CATEGORIA_ADM","SG_UF_CAMPUS","DS_REGIAO_CAMPUS","NO_MUNICIPIO_CAMPUS","NO_CURSO",
             "DS_GRAU","DS_TURNO","QT_VAGAS_CONCORRENCIA","QT_INSCRICAO","CI","IGC","POP","HIERARQUIA_URBANA",
             "COD_HIERARQUIA_URBANA","GRUPO_HIERARQUIA_URBANA","IA","IA_Q5","NU_NOTACORTE"]]
sisu.shape
```

```
(5709, 19)
```

Para ter mais informações gerais dos dados foram utilizadas as funções *dtypes* para identificar o tipo dos atributos e *isnull().sum()* para somar a quantidade de valores nulos nas colunas. Pela Figura 22, observa-se que o conjunto contém 6 variáveis numéricas (*float64* e *int64*) e 13 variáveis categóricas (*object*). Também pode ser visto que não há valores nulos.

Figura 22: Tipos dos atributos e valores nulos no *dataset*

```
sisu.dtypes
```

```
SG_IES                    object
DS_CATEGORIA_ADM          object
SG_UF_CAMPUS              object
DS_REGIAO_CAMPUS          object
NO_MUNICIPIO_CAMPUS       object
NO_CURSO                  object
DS_GRAU                   object
DS_TURNO                  object
QT_VAGAS_CONCORRENCIA      int64
QT_INSCRICAO             float64
CI                        object
IGC                       object
POP                      float64
HIERARQUIA_URBANA         object
COD_HIERARQUIA_URBANA     object
GRUPO_HIERARQUIA_URBANA   object
IA                       float64
IA_Q5                    float64
NU_NOTACORTE             float64
dtype: object
```

```
sisu.isnull().sum()
```

```
SG_IES                    0
DS_CATEGORIA_ADM          0
SG_UF_CAMPUS              0
DS_REGIAO_CAMPUS          0
NO_MUNICIPIO_CAMPUS       0
NO_CURSO                  0
DS_GRAU                   0
DS_TURNO                  0
QT_VAGAS_CONCORRENCIA     0
QT_INSCRICAO              0
CI                        0
IGC                       0
POP                       0
HIERARQUIA_URBANA         0
COD_HIERARQUIA_URBANA     0
GRUPO_HIERARQUIA_URBANA   0
IA                        0
IA_Q5                     0
NU_NOTACORTE              0
dtype: int64
```

Apesar de não serem observados valores faltantes, realizando as primeiras explorações nas variáveis foi visualizado que os índices de qualidade do MEC (IGC e CI) possuem algumas observações preenchidas com “-”, como se não houvesse uma nota calculada nestes casos.

Figura 23: Contagem de valores únicos das variáveis CI e IGC

```
sisu.IGC.value_counts()
```

```
4    4014
3     907
5     784
-       4
Name: IGC, dtype: int64
```

```
sisu.CI.value_counts()
```

```
5    2062
4    1924
-    1373
3     318
2      32
Name: CI, dtype: int64
```

Para sanar estes problemas de valores faltantes optou-se por calcular a moda dos valores de índice dos grupos, a causa da escolha se dá por conta de as universidades serem do estado e como os índices são de qualidade institucional, foi

pensado em generalizar o valor para os casos faltantes, portanto sendo utilizado a categoria do índice que mais foi vista para as universidades da mesma unidade da federação. Para o IGC foi observado apenas uma instituição, com isso foi calculado a moda dos índices de instituições públicas federais do estado do Rio de Janeiro e foi preenchido com este valor. Como visto na Figura 25, foi separado um *dataframe* contendo apenas as informações das IES, já que, na base de dados, cada universidade pode ofertar mais de um curso. O segundo passo foi pegar o valor da moda do grupo preterido e atribuir o valor da métrica as observações necessárias.

Figura 24: Observações com valores de IGC não preenchidos

```
sisu[sisu.IGC=='-']
```

	SG_IES	DS_CATEGORIA_ADM	SG_UF_CAMPUS	DS_REGIAO_CAMPUS	NO_MUNICIPIO_CAMPUS	NO_CURSO	DS_GRAU	DS_TURNO	QT_VAGAS_CON
4843	CP II	Pública Federal	RJ	Sudeste	Rio de Janeiro	HISTÓRIA	Licenciatura	Noturno	
4844	CP II	Pública Federal	RJ	Sudeste	Rio de Janeiro	FILOSOFIA	Licenciatura	Noturno	
4845	CP II	Pública Federal	RJ	Sudeste	Rio de Janeiro	GEOGRAFIA	Licenciatura	Noturno	
4846	CP II	Pública Federal	RJ	Sudeste	Rio de Janeiro	CIÊNCIAS SOCIAIS	Licenciatura	Noturno	

Figura 25: Separação da base de IES's, cálculo e atribuição da moda

```
ies_igc = sisu[['SG_IES','DS_CATEGORIA_ADM','SG_UF_CAMPUS','IGC']].drop_duplicates()
```

```
moda_rj = ies_igc[(ies_igc.DS_CATEGORIA_ADM=='Pública Federal') & (ies_igc.SG_UF_CAMPUS=='RJ')].IGC.mode()[0]
sisu.IGC = sisu.IGC.apply(lambda x: moda_rj if x=='-' else x)
```

```
sisu.IGC.value_counts()
```

```
4    4018
3     907
5     784
Name: IGC, dtype: int64
```

Para o Conceito Institucional, foi observado que mais de uma instituição não teve valores atribuídos. Para a imputação de valores nessas observações também foi escolhido a moda do grupo, porém apenas da UF da IES. O motivo disso foi por

todas as universidades com “-” são as públicas estaduais e municipais, enquanto todas públicas federais tiveram valores preenchidos.

Figura 26: Observações com valores não atribuídos

```
sisu[sisu.CI=='-']
```

	SG_IES	DS_CATEGORIA_ADM	SG_UF_CAMPUS	DS_REGIAO_CAMPUS	NO_MUNICIPIO_CAMPUS	NO_CURSO	DS_GRAU	DS_TURNO	QT_VAGAS
151	UPE	Pública Estadual	PE	Nordeste	Nazaré da Mata	GEOGRAFIA	Licenciatura	Noturno	
152	UPE	Pública Estadual	PE	Nordeste	Nazaré da Mata	HISTÓRIA	Licenciatura	Noturno	
153	UPE	Pública Estadual	PE	Nordeste	Nazaré da Mata	LETRAS - PORTUGUÊS E INGLÊS	Licenciatura	Noturno	
154	UPE	Pública Estadual	PE	Nordeste	Nazaré da Mata	PEDAGOGIA	Licenciatura	Noturno	
155	UPE	Pública Estadual	PE	Nordeste	Nazaré da Mata	CIÊNCIAS BIOLÓGICAS	Licenciatura	Vespertino	
...	...	...	...	...	...	...	...	...	
5704	UNIMONTES	Pública Estadual	MG	Sudeste	Montes Claros	CIÊNCIAS DA RELIGIÃO	Licenciatura	Noturno	
5705	UNIMONTES	Pública Estadual	MG	Sudeste	Montes Claros	CIÊNCIAS SOCIAIS	Bacharelado	Matutino	
5706	UNIMONTES	Pública Estadual	MG	Sudeste	Montes Claros	ENGENHARIA CIVIL	Bacharelado	Integral	
5707	UNIMONTES	Pública Estadual	MG	Sudeste	Montes Claros	GEOGRAFIA	Bacharelado	Matutino	
5708	UNIMONTES	Pública Estadual	MG	Sudeste	Montes Claros	FÍSICA	Licenciatura	Noturno	

1373 rows × 19 columns

Figura 27: Características das instituições com valores faltantes

```
sisu.DS_CATEGORIA_ADM.value_counts()
```

```
Pública Federal      4468
Pública Estadual     1240
Pública Municipal       1
Name: DS_CATEGORIA_ADM, dtype: int64
```

```
sisu[sisu.CI=='-'].DS_CATEGORIA_ADM.value_counts()
```

```
Pública Estadual     1240
Pública Federal       132
Pública Municipal       1
Name: DS_CATEGORIA_ADM, dtype: int64
```

Foram realizadas as mesmas etapas descritas anteriormente, separando um *dataframe* com os valores únicos das instituições e seus respectivos CI’s e depois foi

agrupado pela unidade federativa das IES's e calculada a moda. Após obter as modas UF's foi realizado a operação de *join* para gerar uma coluna com a moda do índice. Por último foi criado uma função para atribuir o valor da métrica nas observações preenchidas com "-" e retirada a coluna da moda com a função *drop*.

Figura 28: Etapas da imputação de valores na variável CI

```
ies_ci = sisu[sisu.CI!='-'][['SG_IES','SG_UF_CAMPUS','CI']].drop_duplicates()
```

```
# Criar a tabela com a moda da coluna 'CI' agrupada pela UF
modas_CI = ies_ci.groupby('SG_UF_CAMPUS')['CI'].apply(lambda x: x.mode().iloc[0] if not x.mode().empty else None).reset_index()

# Renomear a coluna 'CI' para 'MODA_CI'
modas_CI.rename(columns={'CI': 'MODA_CI'}, inplace=True)

# Fazer o join com o DataFrame sisu
sisu = sisu.merge(modas_CI, on='SG_UF_CAMPUS', how='left')

# Substituir valores '-' em 'CI' pela moda
sisu['CI'] = sisu.apply(lambda row: row['MODA_CI'] if row['CI'] == '-' else row['CI'], axis=1)

# Remover a coluna 'MODA_CI' se não for mais necessária
sisu.drop(columns=['MODA_CI'], inplace=True)
```

```
sisu.CI.value_counts()
```

```
4    3162
5    2142
3     373
2      32
Name: CI, dtype: int64
```

O último tratamento realizado nas variáveis foi no grupo da hierarquia urbana. Observou-se que havia algumas subdivisões nos grupos, e optou-se por generalizar a informação, mantendo apenas o grupo principal da hierarquia urbana. Exemplo: "1 – Metrópole – Integrante de Arranjo Populacional" e "1 - Metrópole" foi resumido em um único grupo "1", que é denominado às metrópoles.

Figura 29: Tratamento da variável Grupo da Hierarquia Urbana

```
sisu.GRUPO_HIERARQUIA_URBANA.value_counts()
```

```
1 - Metrópole - Integrante de Arranjo Populacional             2291
2 - Capital Regional - Integrante de Arranjo Populacional      1726
2 - Capital Regional                                            870
3 - Centro Sub-Regional                                         372
3 - Centro Sub-Regional - Integrante de Arranjo Populacional    314
1 - Metrópole                                                   115
5 - Centro Local - Integrante de Arranjo Populacional            20
5 - Centro Local                                                  1
Name: GRUPO_HIERARQUIA_URBANA, dtype: int64
```

```
sisu.GRUPO_HIERARQUIA_URBANA = sisu.GRUPO_HIERARQUIA_URBANA.apply(lambda x: str(x)[0])
sisu.GRUPO_HIERARQUIA_URBANA.value_counts()
```

```
2    2596
1    2406
3     686
5      21
Name: GRUPO_HIERARQUIA_URBANA, dtype: int64
```

4. Análise e Exploração dos Dados

A análise exploratória tem como objetivo identificar padrões e relações importantes entre as variáveis que podem influenciar as notas de corte dos cursos oferecidos pelo SISU. Nesta etapa, foram examinadas as distribuições das variáveis, correlações e tendências, buscando levantar hipóteses que ajudem a entender melhor os fatores determinantes para as notas dos cursos. A seguir, são apresentados os resultados e insights obtidos a partir de visualizações gráficas, análises estatísticas e tratamentos realizados.

Para realizar a análise foram importadas algumas bibliotecas de visualização, de processamento e de modelagem de dados.

Figura 30: Importação de Bibliotecas e definição da base

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
```

```
data = sisu.copy()
```

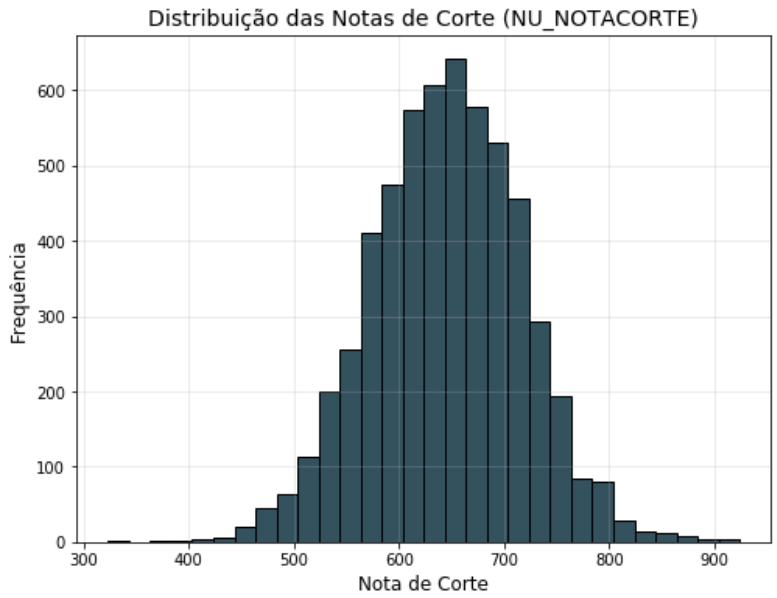
```
# Definir cores suaves
cor_histograma = '#34515e'  # Azul escuro suave
cor_boxplot = '#ffc04d'  # Amarelado mais escuro
```

4.1. Distribuição das Notas de Corte

O histograma, na Figura 31, mostra uma distribuição próxima a uma distribuição normal, com a maioria das notas de corte concentradas ao redor da média, entre 600 e 700 pontos, e poucas acima de 800 ou abaixo de 500. Isso indica que a maior parte dos cursos possui uma competitividade moderada e similar, enquanto um pequeno grupo de cursos, como Medicina e Engenharia, apresentam notas mais elevadas, enquanto História e Pedagogia notas menores.

Figura 31: Geração de Histograma e Tabela da Distribuição de Notas de Corte

```python
# Gerar o histograma para a distribuição das notas de corte (NU_NOTACORTE)
plt.figure(figsize=(8, 6))
plt.hist(data['NU_NOTACORTE'], bins=30, color=cor_histograma, edgecolor='black')
plt.title('Distribuição das Notas de Corte (NU_NOTACORTE)', fontsize=14)
plt.xlabel('Nota de Corte', fontsize=12)
plt.ylabel('Frequência', fontsize=12)
plt.grid(True, alpha=0.3)
plt.show()

# Gerar tabela resumo com média, mediana, desvio padrão, valor mínimo e máximo das notas de corte
nota_corte_resumo = data['NU_NOTACORTE'].describe().to_frame().T
nota_corte_resumo['mediana'] = data['NU_NOTACORTE'].median()
nota_corte_resumo
```



	count	mean	std	min	25%	50%	75%	max	mediana
NU_NOTACORTE	5709.0	644.579136	71.431199	323.18	596.15	645.7	693.86	923.9	645.7

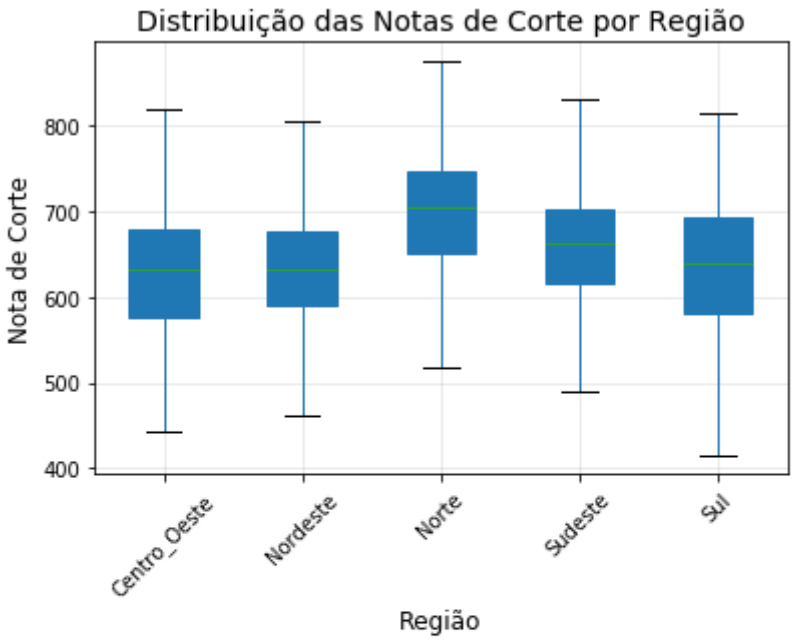
4.2. Notas de Corte por Região

O boxplot revela que a região Norte, ao contrário do esperado, possui uma distribuição de notas de corte elevada. O Sudeste apresenta valores relativamente altos, enquanto Nordeste e Sul possuem médias muito próximas, fato já esperado devido ao aumento de renome das universidades nordestinas e, por último, o Centro-Oeste que possui notas mais baixas. Isso sugere que a demanda e a disponibilidade de vagas podem estar influenciando nas variações regionais.

Figura 32: Script para geração de Boxplot e Tabela das Notas de Corte por Região

```
# Gerar o boxplot para as notas de corte por região (DS_REGIAO_CAMPUS)
plt.figure(figsize=(10, 6))
data.boxplot(column='NU_NOTACORTE', by='DS_REGIAO_CAMPUS', grid=False, showfliers=False,
             patch_artist=True, boxprops=dict(facecolor=cor_boxplot))
plt.title('Distribuição das Notas de Corte por Região', fontsize=14)
plt.suptitle('')  # Remover o título adicional gerado pelo boxplot
plt.xlabel('Região', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, alpha=0.3)
plt.show()

# Gerar a tabela com a média das notas de corte por região
media_notas_regiao = data.groupby('DS_REGIAO_CAMPUS')['NU_NOTACORTE'].mean().reset_index()
media_notas_regiao
```

Figura 33: Boxplot e Tabela das Notas de Corte por Região



	DS_REGIAO_CAMPUS	NU_NOTACORTE
0	Centro_Oeste	627.173381
1	Nordeste	636.096243
2	Norte	696.941009
3	Sudeste	658.260916
4	Sul	634.135610

4.3. Notas de Corte por Estado

A análise por estado mostra que Amazonas (AM), Amapá (AP) e Pará (PA) possuem notas de corte mais altas, enquanto Mato Grosso (MT) e Roraima (RR) apresentam as mais baixas. Esse resultado, exibido na Tabela da Figura 36, reflete variações na competitividade entre os estados, podemos observar essa variação na distribuição de valores no Boxplot da Figura 35.

Figura 34: Script para geração de Boxplot e Tabela das Notas de Corte por Estado

```
# Gerar o boxplot para as notas de corte por estado (SG_UF_CAMPUS)
plt.figure(figsize=(12, 8))
data.boxplot(column='NU_NOTACORTE', by='SG_UF_CAMPUS', grid=False, showfliers=False, patch_artist=True,
             boxprops=dict(facecolor=cor_boxplot))
plt.title('Distribuição das Notas de Corte por Estado', fontsize=14)
plt.suptitle('')  # Remover o título adicional gerado pelo boxplot
plt.xlabel('Estado', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.xticks(rotation=90)
plt.grid(True, alpha=0.3)
plt.show()

# Gerar a tabela com a média das notas de corte por estado
media_notas_estado = data.groupby('SG_UF_CAMPUS')['NU_NOTACORTE'].mean().reset_index()
media_notas_estado.sort_values('NU_NOTACORTE', ascending=False)
```

Figura 35: Boxplot das Notas de Corte por Estado

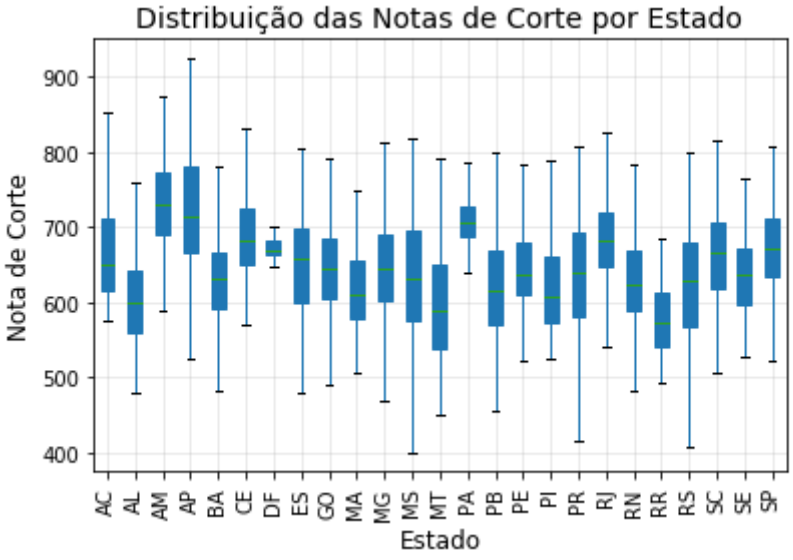


Figura 36: Tabela das Notas de Corte médias por Estado

SG_UF_CAMPUS	NU_NOTACORTE
AM	732.024696
PA	710.511951
AP	710.481429
CE	690.100090
RJ	680.290023
DF	676.450833
SP	670.168173
AC	662.382051
SC	657.553066
ES	651.527583
PE	650.288353
MG	645.742685
GO	643.447164
SE	639.357179
PR	632.938079
RN	632.688739
MS	632.019694
BA	630.889030
RS	624.071265
PB	622.811031
MA	621.840094
PI	618.426118
AL	607.434161
MT	596.325705
RR	584.932308

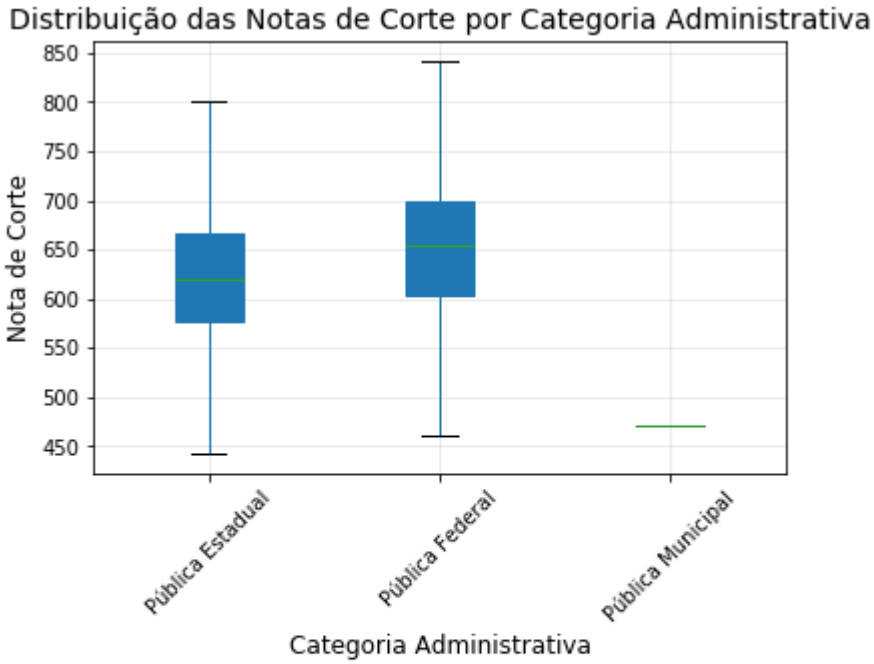
4.4. Notas de Corte por Categoria Administrativa

O Boxplot confirma que instituições públicas federais apresentam as maiores notas de corte, enquanto as públicas municipais, embora em menor quantidade de dados, mostram valores bem inferiores. As instituições públicas estaduais têm uma variação maior, com algumas notas entre 800 e 450 pontos, conforme mostrado na tabela e gráfico das figuras abaixo.

Figura 37: Script para geração de Boxplot e Tabela das Notas de Corte por Categoria Administrativa

```
# Gerar o boxplot para as notas de corte por categoria administrativa (DS_CATEGORIA_ADM)
plt.figure(figsize=(8, 6))
data.boxplot(column='NU_NOTACORTE', by='DS_CATEGORIA_ADM', grid=False, showfliers=False, patch_artist=True,
             boxprops=dict(facecolor=cor_boxplot))
plt.title('Distribuição das Notas de Corte por Categoria Administrativa', fontsize=14)
plt.suptitle('')  # Remover o título adicional gerado pelo boxplot
plt.xlabel('Categoria Administrativa', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, alpha=0.3)
plt.show()

# Gerar a tabela com a média das notas de corte por categoria administrativa
media_notas_categoria = data.groupby('DS_CATEGORIA_ADM')['NU_NOTACORTE'].mean().reset_index()
media_notas_categoria
```

Figura 38: Boxplot e Tabela das Notas de Corte por Categoria Administrativa



	DS_CATEGORIA_ADM	NU_NOTACORTE
0	Pública Estadual	622.707839
1	Pública Federal	650.688109
2	Pública Municipal	470.100000

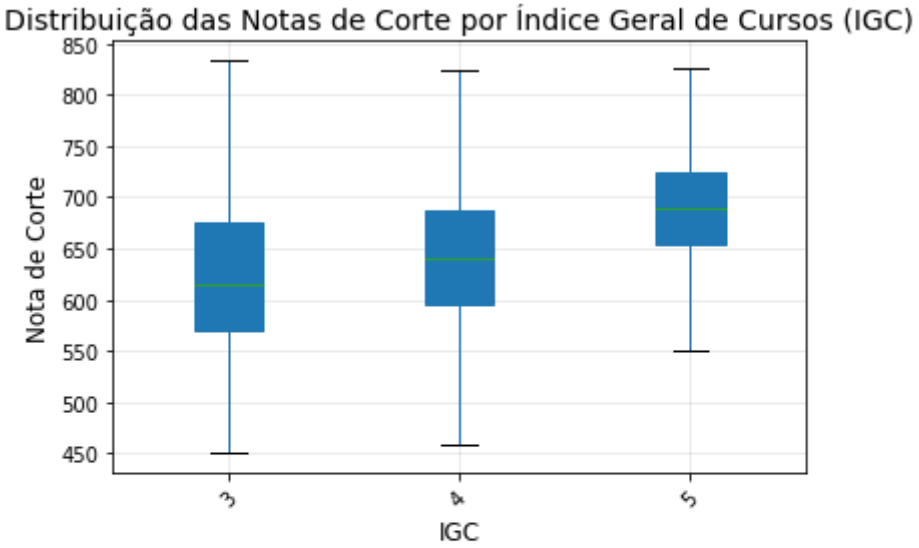
4.5. Notas de Corte por Índice Geral de Cursos (IGC)

A tabela e o Boxplot revelam que universidades com índices maiores possuem em média maiores notas de corte, porém pelo Boxplot mostra uma faixa de valores semelhantes entre a métrica institucional. O IGC, por si só, talvez não seja um fator determinante para as notas de corte.

Figura 39: Script para geração de Boxplot e Tabela das Notas de Corte por IGC

```python
# Gerar o boxplot para as notas de corte por categoria administrativa (DS_CATEGORIA_ADM)
plt.figure(figsize=(8, 6))
data.boxplot(column='NU_NOTACORTE', by='IGC', grid=False, showfliers=False, patch_artist=True,
             boxprops=dict(facecolor=cor_boxplot))
plt.title('Distribuição das Notas de Corte por Índice Geral de Cursos (IGC)', fontsize=14)
plt.suptitle('')  # Remover o título adicional gerado pelo boxplot
plt.xlabel('IGC', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, alpha=0.3)
plt.show()

# Gerar a tabela com a média das notas de corte por categoria administrativa
media_notas_categoria = data.groupby('IGC')['NU_NOTACORTE'].mean().reset_index()
media_notas_categoria
```

Figura 40: Boxplot e Tabela das Notas de Corte por IGC



	IGC	NU_NOTACORTE
0	3	623.900110
1	4	641.088945
2	5	686.389681

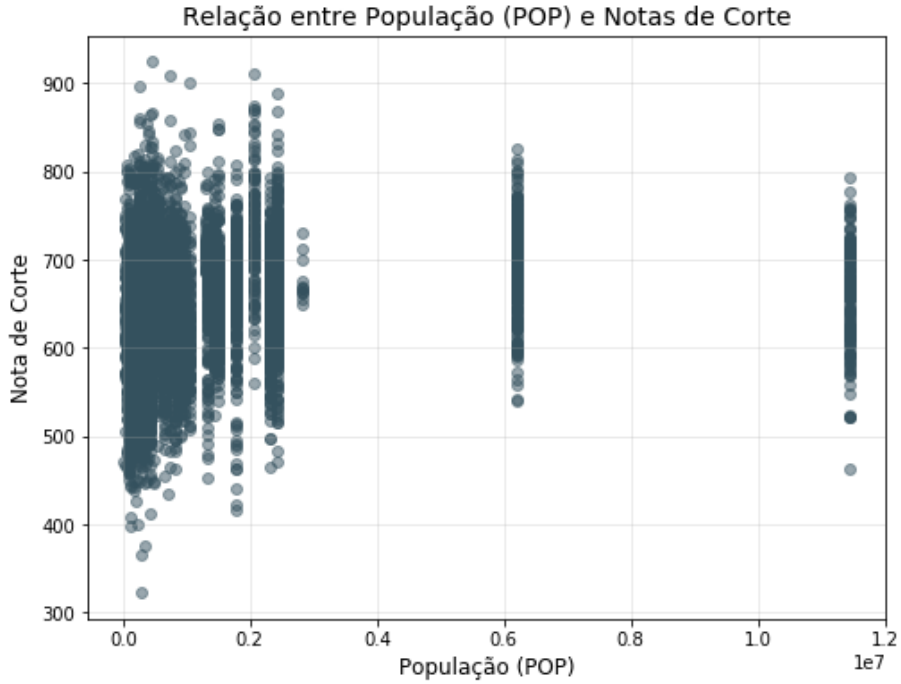
4.6. Notas de Corte por População

O coeficiente de Pearson mostra uma correlação fraca (0.16) entre a população do município e as notas de corte dos cursos, o gráfico na Figura 41

mostra uma distribuição um pouco aleatória entre as variáveis. Embora a população tenha algum efeito, outros fatores como o prestígio da instituição e demanda por cursos específicos parecem ser mais relevantes, conforme demonstrado nos gráficos anteriores.

Figura 41: Geração de Gráfico de Dispersão e Correlação das Notas de Corte pela População

```
# Gerar o gráfico de dispersão (scatter plot) para a relação entre POP e NU_NOTACORTE
plt.figure(figsize=(8, 6))
plt.scatter(data['POP'], data['NU_NOTACORTE'], alpha=0.5, color=cor_histograma)
plt.title('Relação entre População (POP) e Notas de Corte', fontsize=14)
plt.xlabel('População (POP)', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.grid(True, alpha=0.3)
plt.show()

# Calcular a correlação entre POP e NU_NOTACORTE
correlacao_pop_nota = data[['POP', 'NU_NOTACORTE']].corr()
correlacao_pop_nota
```



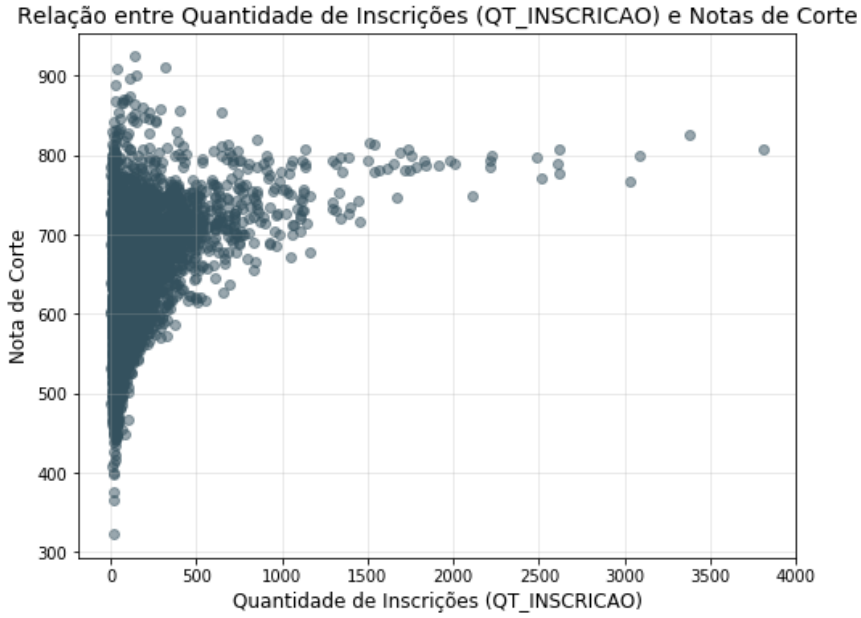
	POP	NU_NOTACORTE
POP	1.000000	0.155786
NU_NOTACORTE	0.155786	1.000000

4.7. Notas de Corte por Quantidade de Inscrições

Com uma correlação de 0.40, a quantidade de inscrições tem uma relação moderada com as notas de corte, indicando que cursos com maior número de inscritos tendem a ser mais competitivos, obtendo notas de corte mais altas. Isso é mostrado na tabela de correlação e no gráfico de dispersão abaixo observa-se uma curva suavizada sendo possível visualizar essa relação positiva entre as variáveis.

Figura 42: Geração de Gráfico de Dispersão e Correlação das Notas de Corte pela Quantidade de Inscrições

```
# Gerar o gráfico de dispersão (scatter plot) para a relação entre QT_INSCRICAO e NU_NOTACORTE
plt.figure(figsize=(8, 6))
plt.scatter(data['QT_INSCRICAO'], data['NU_NOTACORTE'], alpha=0.5, color=cor_histograma)
plt.title('Relação entre Quantidade de Inscrições (QT_INSCRICAO) e Notas de Corte', fontsize=14)
plt.xlabel('Quantidade de Inscrições (QT_INSCRICAO)', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.grid(True, alpha=0.3)
plt.show()

# Calcular a correlação entre QT_INSCRICAO e NU_NOTACORTE
correlacao_inscricao_nota = data[['QT_INSCRICAO', 'NU_NOTACORTE']].corr()
correlacao_inscricao_nota
```



	QT_INSCRICAO	NU_NOTACORTE
QT_INSCRICAO	1.000000	0.397343
NU_NOTACORTE	0.397343	1.000000

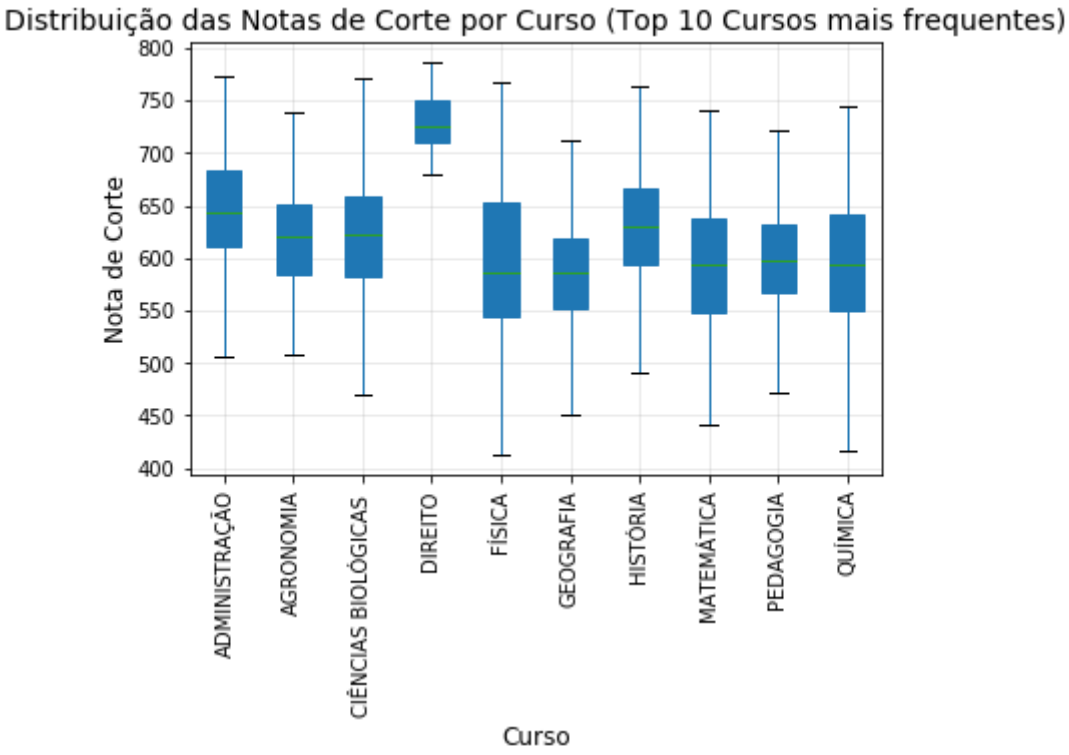
4.8. Notas de Corte por Curso de Graduação

O Boxplot, na Figura 44, mostra que alguns cursos, como Direito, possuem competitividade muito alta, independente da instituição, enquanto outros como Química, Física e Matemática encontram uma grande variação entre as notas de corte. Pela tabela, confirma-se que cursos como Engenharia Mecatrônica e Medicina possuem notas de corte consideravelmente mais altas, obtendo uma média de 50 a 70 pontos maior que a décima maior nota de corte média. Isso reflete a alta demanda por cursos de maior prestígio no mercado de trabalho e uma possível boa variável preditora.

Figura 43: Script para geração de Boxplot e Tabela das Notas de Corte por Curso

```
# Gerar o boxplot para as notas de corte por curso (NO_CURSO)
cursos_mais_frequentes = data['NO_CURSO'].value_counts().nlargest(10).index  # Selecionar os 10 cursos mais frequentes
data_filtrada = data[data['NO_CURSO'].isin(cursos_mais_frequentes)]  # Filtrar os 10 cursos mais frequentes

plt.figure(figsize=(12, 8))
data_filtrada.boxplot(column='NU_NOTACORTE', by='NO_CURSO', grid=False, showfliers=False, patch_artist=True,
                      boxprops=dict(facecolor=cor_boxplot))
plt.title('Distribuição das Notas de Corte por Curso (Top 10 Cursos mais frequentes)', fontsize=14)
plt.suptitle('')  # Remover o título adicional gerado pelo boxplot
plt.xlabel('Curso', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.xticks(rotation=90)
plt.grid(True, alpha=0.3)
plt.show()

# Gerar tabela resumo com as 10 maiores média das notas de corte para os cursos
media_notas_curso = data.groupby('NO_CURSO')['NU_NOTACORTE'].mean().nlargest(10).reset_index()
media_notas_curso
```

Figura 44: Boxplot e Tabela das Notas de Corte por Curso



	NO_CURSO	NU_NOTACORTE
0	ENGENHARIA DE MECATRÔNICA	831.720000
1	MEDICINA	810.790247
2	ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO	809.210000
3	ENGENHARIA AEROESPACIAL	782.860000
4	ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO	781.210000
5	INTELIGÊNCIA ARTIFICIAL	778.810000
6	ENGENHARIA NUCLEAR	770.460000
7	ENGENHARIA AERONÁUTICA	770.455000
8	CINEMA	764.450000
9	ENGENHARIA DA COMPUTAÇÃO	763.624444

4.9. Notas de Corte por Nível de Hierarquia Urbana

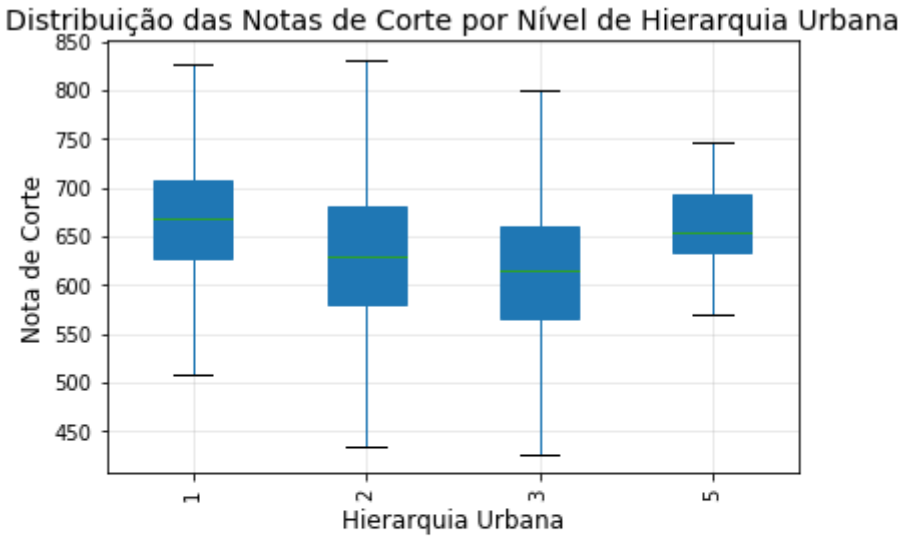
O Boxplot e tabela, na Figura 46, revela que metrópoles (1) e capitais regionais (2) possuem notas de corte mais altas, enquanto cidades com menor hierarquia urbana tendem a ter notas mais baixas. O menor nível de hierarquia (5 – Centro Local) possui uma média mais alta, porém observa-se pelo Boxplot que a

variação de valores é pequena, isso pode ter sido resultado da baixa quantidade de municípios com esta hierarquia, vinte e um (21). Esse resultado sugere que a centralidade urbana afeta a competitividade dos cursos.

Figura 45: Script para geração de Boxplot e Tabela das Notas de Corte por Hierarquia Urbana

```
# Gerar o boxplot para as notas de corte por nível de hierarquia urbana
plt.figure(figsize=(10, 6))
data.boxplot(column='NU_NOTACORTE', by='GRUPO_HIERARQUIA_URBANA', grid=False, showfliers=False, patch_artist=True,
             boxprops=dict(facecolor=cor_boxplot))
plt.title('Distribuição das Notas de Corte por Nível de Hierarquia Urbana', fontsize=14)
plt.suptitle('')  # Remover o título adicional gerado pelo boxplot
plt.xlabel('Hierarquia Urbana', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.xticks(rotation=90)
plt.grid(True, alpha=0.3)
plt.show()

# Gerar a tabela com a média das notas de corte por grupo de hierarquia urbana
media_notas_hierarquia = data.groupby('GRUPO_HIERARQUIA_URBANA')['NU_NOTACORTE'].mean().reset_index()
media_notas_hierarquia
```

Figura 46: Boxplot e Tabela das Notas de Corte por Hierarquia Urbana



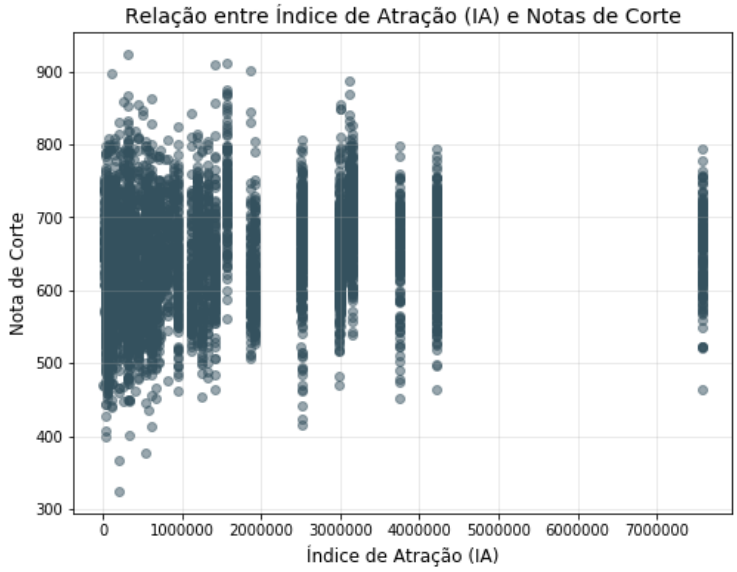
	GRUPO_HIERARQUIA_URBANA	NU_NOTACORTE
0	1	666.818130
1	2	632.403082
2	3	612.557070
3	5	647.872381

4.10. Notas de Corte por Índice de Atração Geral

O gráfico de dispersão mostra uma correlação positiva (0.15) entre o Índice de Atração e as notas de corte, indicando que municípios com maior índice de atração tendem a ter notas de corte mais altas, conforme mostrado na Figura 47.

Figura 47: Geração de Gráfico de Dispersão e Correlação das Notas de Corte pela Atração Geral

```
# Gerar o gráfico de dispersão (scatter plot) para a relação entre IA (Índice de Atração) e NU_NOTACORTE
plt.figure(figsize=(8, 6))
plt.scatter(data['IA'], data['NU_NOTACORTE'], alpha=0.5, color=cor_histograma)
plt.title('Relação entre Índice de Atração (IA) e Notas de Corte', fontsize=14)
plt.xlabel('Índice de Atração (IA)', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.grid(True, alpha=0.3)
plt.show()

# Calcular a correlação entre IA e NU_NOTACORTE
correlacao_ia_nota = data[['IA', 'NU_NOTACORTE']].corr()
correlacao_ia_nota
```



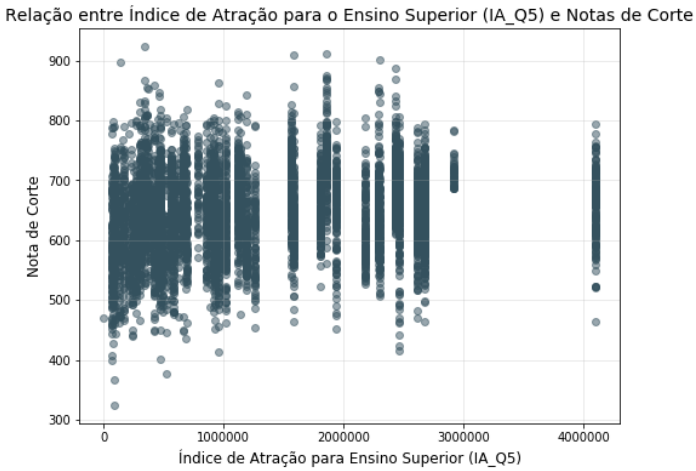
	IA	NU_NOTACORTE
IA	1.000000	0.155273
NU_NOTACORTE	0.155273	1.000000

4.11. Notas de Corte por Índice de Atração para o Ensino Superior

Com uma correlação positiva de 0.16, o índice de atração para o ensino superior também demonstra uma baixa relação com as notas de corte, porém ainda sugere que municípios com maior atração para estudantes de ensino superior tendem a ter maior competitividade por vagas.

Figura 48: Geração de Gráfico de Dispersão e Correlação das Notas de Corte pela Atração para o Ensino Superior

```
# Gerar o gráfico de dispersão (scatter plot) para a relação entre IA_Q5 (Índice de Atração para Ensino Superior) e NU_NOTACORTE
plt.figure(figsize=(8, 6))
plt.scatter(data['IA_Q5'], data['NU_NOTACORTE'], alpha=0.5, color=cor_histograma)
plt.title('Relação entre Índice de Atração para o Ensino Superior (IA_Q5) e Notas de Corte', fontsize=14)
plt.xlabel('Índice de Atração para Ensino Superior (IA_Q5)', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.grid(True, alpha=0.3)
plt.show()

# Calcular a correlação entre IA_Q5 e NU_NOTACORTE
correlacao_iaq5_nota = data[['IA_Q5', 'NU_NOTACORTE']].corr()
correlacao_iaq5_nota
```



	IA_Q5	NU_NOTACORTE
IA_Q5	1.000000	0.157931
NU_NOTACORTE	0.157931	1.000000

4.12. Target Encoding na variável Nome do Curso

Devido à presença de muitos cursos diferentes (507), se torna inviável a aplicação de *features* para todos. Como foi visto na análise exploratória, este atributo parece ter considerável influência nas notas, portanto, para contornar o problema foi decidido aplicar a técnica de ***Target Encoding com Smoothing (suavização)***. A abordagem é robusta e serve para codificar os nomes de curso em apenas uma única *feature* de forma mais eficiente, por conta de graduações com poucas observações no conjunto de dados foi necessário aplicar o *smoothing* para ajudar a reduzir o impacto de categorias com poucos dados, suavizando as médias dos cursos raros em direção à média global, evitando *overfitting*.

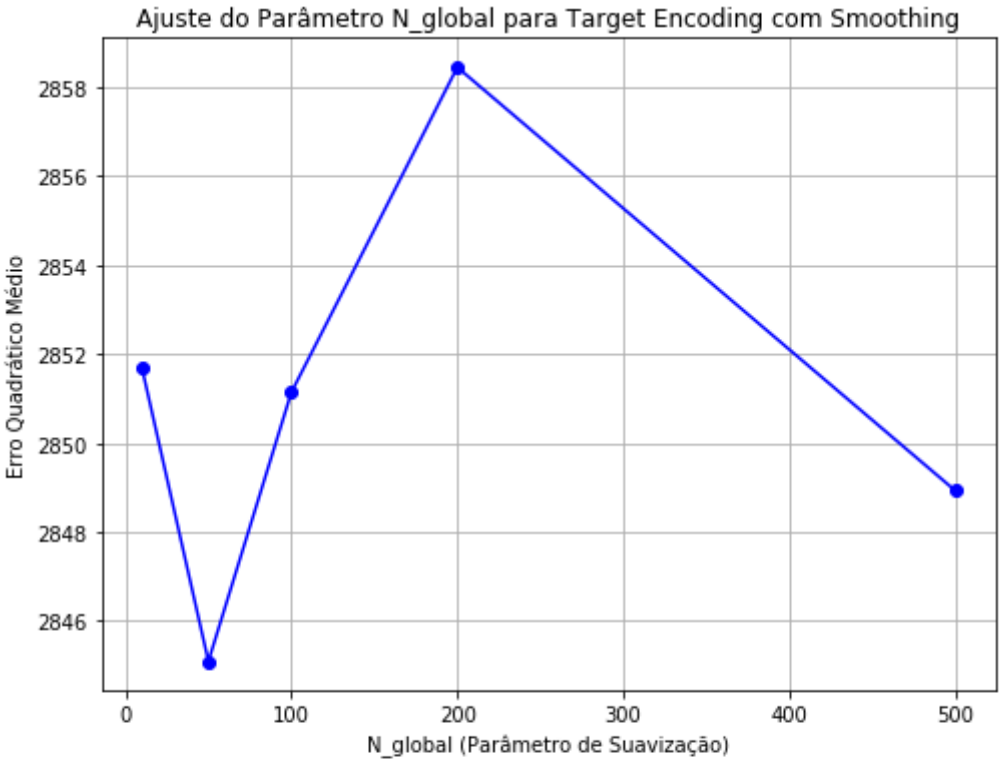
A suavização é essencial para cursos com poucas observações, evitando que eles dominem a codificação com valores extremos. O valor suavizado pondera a média global das notas de corte com a média específica de cada curso, ajustando a influência conforme o número de observações. Cursos com muitas observações mantêm sua média específica, enquanto aqueles com menos dados são mais influenciados pela média global. Logo, ao invés de ser uma variável categórica com o nome do curso o atributo é transformado em uma variável numérica contendo a média suavizada pela técnica para cada graduação, oferecendo um equilíbrio entre capturar a variabilidade dos cursos e evitar overfitting, especialmente em cursos com poucos dados.

Para aplicar a técnica é necessário estimar um parâmetro de suavização, para isso foi estimada uma *Random Forest* utilizando a nota de corte como variável predita e o curso aplicado o *target encoding* como preditora. Por último observa-se a métrica do erro quadrático médio (EQM) e foi escolhido o valor com menor EQM. Ao testar diferentes valores para o parâmetro, identificou-se que **N_global = 50** foi o valor ideal, resultando na menor taxa de erro quadrático médio, conforme na Figura 49. Esse valor permitiu manter a robustez do modelo, preservando as diferenças significativas entre os cursos sem exagerar nas variações extremas.

Figura 49: Script para aplicação de Target Encoding e geração de Gráfico para encontrar melhor parâmetro de suavização

```python
# Função para aplicar target encoding com smoothing e calcular o desempenho do modelo
def calcular_desempenho(N_global):
    # Calcular a média global das notas de corte
    media_global = data['NU_NOTACORTE'].mean()

    # Calcular a média de notas de corte por curso e o número de observações de cada curso
    curso_stats = data.groupby('NO_CURSO')['NU_NOTACORTE'].agg(['mean', 'count']).reset_index()
    curso_stats.columns = ['NO_CURSO', 'media_curso', 'contagem_curso']

    # Aplicar o smoothing
    curso_stats['curso_smoothed'] = (curso_stats['contagem_curso'] * curso_stats['media_curso'] +
                                     N_global * media_global) / (curso_stats['contagem_curso'] + N_global)

    # Mapear os valores suavizados de volta para a base de dados original
    data_smooth = data.merge(curso_stats[['NO_CURSO', 'curso_smoothed']], on='NO_CURSO', how='left')

    # Separar as features e o target
    X = data_smooth[['curso_smoothed']]  # Neste exemplo, estamos usando apenas o encoding dos cursos como feature
    y = data_smooth['NU_NOTACORTE']

    # Treinar um modelo simples (Random Forest) para avaliar o desempenho com diferentes valores de N_global
    modelo = RandomForestRegressor(n_estimators=100, random_state=42)

    # Avaliar o desempenho usando validação cruzada (com 5 folds)
    scores = cross_val_score(modelo, X, y, cv=5, scoring='neg_mean_squared_error')
    return -scores.mean()  # Retornar o erro quadrático médio (negativo revertido para positivo)


# Definir uma lista de valores de N_global para testar
valores_N_global = [10, 50, 100, 200, 500]

# Testar cada valor de N_global e armazenar os resultados
resultados = []
for N in valores_N_global:
    desempenho = calcular_desempenho(N)
    resultados.append((N, desempenho))

# Converter resultados para um DataFrame para facilitar a visualização
resultados_df = pd.DataFrame(resultados, columns=['N_global', 'Erro Quadrático Médio'])

# Plotar os resultados
plt.figure(figsize=(8, 6))
plt.plot(resultados_df['N_global'], resultados_df['Erro Quadrático Médio'], marker='o', color='b')
plt.title('Ajuste do Parâmetro N_global para Target Encoding com Smoothing')
plt.xlabel('N_global (Parâmetro de Suavização)')
plt.ylabel('Erro Quadrático Médio')
plt.grid(True)
plt.show()

# Mostrar os resultados para avaliação
resultados_df
```

Figura 50: Gráfico de ajuste de parâmetro e Tabela com erros quadráticos médios por N_global



	N_global	Erro Quadrático Médio
0	10	2851.679772
1	50	2845.088372
2	100	2851.140402
3	200	2858.452682
4	500	2848.929567

Embora o ***target encoding com smoothing*** seja uma técnica eficaz, ele apresenta algumas desvantagens. A principal delas é a dependência da escolha do parâmetro de suavização (**N_global**), pois um valor inadequado pode causar *overfitting*, se for muito baixo, ou *underfitting*, se for muito alto, resultando em um modelo preditivo ruim. Além disso, essa técnica pode introduzir vieses em categorias raras com a suavização demais de características específicas. Também é uma abordagem computacionalmente mais custosa e exige cuidado para evitar *leakage* de dados, o que pode ser mitigado aplicando o *encoding* corretamente com validação cruzada, que foi exatamente o que foi realizado na etapa de estimação do parâmetro.

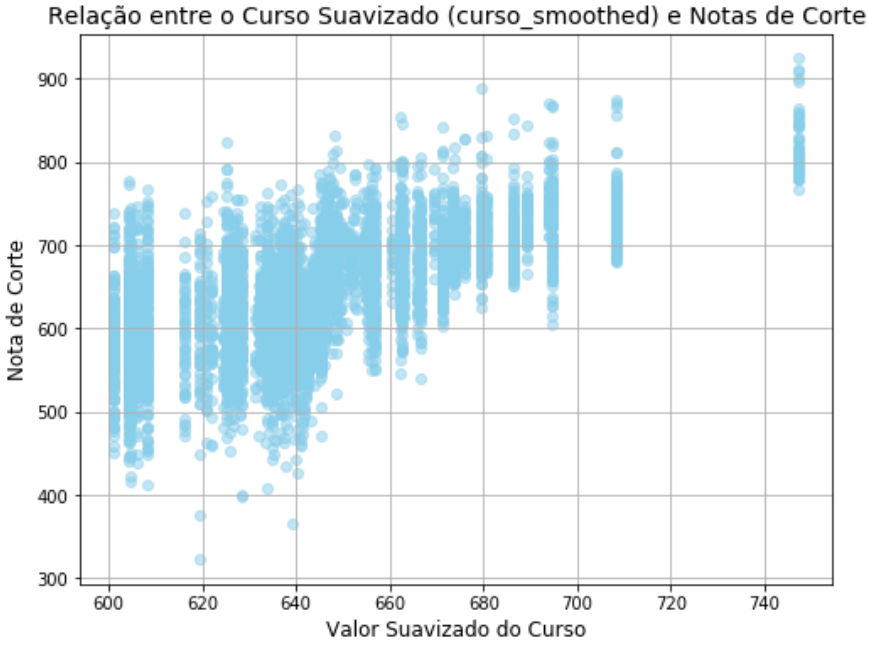
Conforme é possível observar na Figura 52, a aplicação do ***target encoding com smoothing*** nos cursos demonstrou uma clara correlação positiva entre o valor médio das notas dos cursos suavizado e as notas de corte. Como esperado, cursos mais concorridos apresentaram valores suavizados mais elevados e, consequentemente, notas de corte maiores, refletindo sua alta demanda e prestígio. Por outro lado, cursos de menor popularidade obtiveram valores suavizados mais baixos, o que resultou em notas de corte correspondentes menores. O gráfico confirma que à medida que o valor suavizado aumenta, as notas de corte também aumentam, indicando que o ***target encoding com smoothing*** foi eficaz em capturar a competitividade dos cursos sem inflar a dimensionalidade. Além disso, a suavização parece ter evitado a criação de outliers extremos, especialmente em cursos com poucas observações, garantindo robustez ao modelo e prevenindo *overfitting*. Essa abordagem pode agora ser utilizada de forma mais confiável no modelo de machine learning, preservando as diferenças entre os cursos, capturando de maneira eficaz sua competitividade e mantendo a capacidade de generalização do modelo preditivo.

Figura 51: Script para aplicação de Target Encoding com Smoothing nos Cursos

```
N_global = 50

# Calcular a média global das notas de corte
media_global = data['NU_NOTACORTE'].mean()

# Calcular a média de notas de corte por curso e o número de observações de cada curso
curso_stats = data.groupby('NO_CURSO')['NU_NOTACORTE'].agg(['mean', 'count']).reset_index()
curso_stats.columns = ['NO_CURSO', 'media_curso', 'contagem_curso']

# Aplicar o smoothing
curso_stats['curso_smoothed'] = (curso_stats['contagem_curso'] * curso_stats['media_curso'] +
                                 N_global * media_global) / (curso_stats['contagem_curso'] + N_global)

# Mapear os valores suavizados de volta para a base de dados original
data = data.merge(curso_stats[['NO_CURSO', 'curso_smoothed']], on='NO_CURSO', how='left')
```

Figura 52: Geração de Gráfico de Dispersão e Correlação das Notas de Corte pelos Cursos aplicado Target Encoding com suavização

```
# Gerar o scatter plot para mostrar a relação entre o curso suavizado e a nota de corte
plt.figure(figsize=(8, 6))
plt.scatter(data['curso_smoothed'], data['NU_NOTACORTE'], alpha=0.5, color='skyblue')
plt.title('Relação entre o Curso Suavizado (curso_smoothed) e Notas de Corte', fontsize=14)
plt.xlabel('Valor Suavizado do Curso', fontsize=12)
plt.ylabel('Nota de Corte', fontsize=12)
plt.grid(True)
plt.show()

# Calcular a correlação entre IA_Q5 e NU_NOTACORTE
correlacao_curso_smooth_nota = data[['curso_smoothed', 'NU_NOTACORTE']].corr()
correlacao_curso_smooth_nota
```



	curso_smoothed	NU_NOTACORTE
curso_smoothed	1.000000	0.630677
NU_NOTACORTE	0.630677	1.000000

4.13. Conclusão da Análise Exploratória

A análise exploratória forneceu insights valiosos sobre os fatores que influenciam as notas de corte dos cursos oferecidos pelo SISU. A quantidade de inscrições, os cursos aplicados ***target encoding com suavização***, a categoria administrativa e a hierarquia urbana dos municípios das instituições foram identificados como os principais fatores que afetam diretamente as notas de corte,

com destaque para as instituições públicas federais e as metrópoles, que mostraram mais notas de corte competitivas e aos maiores valores suavizados dos cursos e quantidade de inscrições, que tendem a aumentar a nota de corte das graduações.

A aplicação de ***target encoding com smoothing*** foi essencial para lidar com cursos de baixa frequência evitando a inflação da dimensionalidade, permitindo que o modelo consiga capturar a competitividade evitando a influência de valores extremos e possibilitando boa capacidade de generalização para o modelo.

Apesar da fraca correlação entre o Índice Geral de Cursos (IGC) e população com as notas de corte, a análise revelou que a competitividade dos cursos é possivelmente influenciada por fatores geográficos e demográficos, como a localização do município e a atratividade da cidade para estudantes, com o índice de atração para o ensino superior (IA_Q5) e a região da instituição emergindo como preditores razoáveis para as notas dos cursos. A região se mostra preferível de ser utilizada como preditora ao invés do Estado do campus da universidade, pensando também em criar 5 vezes menos features diante da utilização da UF.

Em resumo, a análise exploratória estabeleceu uma base sólida para o desenvolvimento do modelo preditivo das notas de corte, garantindo que variáveis relevantes fossem identificadas e que a codificação dos cursos fosse tratada de maneira adequada para maximizar a precisão do modelo. Esses insights e técnicas serão fundamentais para o sucesso das previsões no modelo final.

5. Criação de Modelos de Machine Learning

5.1. Contextualização do Problema

Prever as notas de corte, do Sistema de Seleção Unificada (SISU), dos cursos em universidades públicas, é essencial para o planejamento e a tomada de decisões de estudantes, instituições e órgãos governamentais. Essas previsões são cruciais para ajudar os candidatos a se prepararem adequadamente para o processo seletivo, permitindo melhor compreensão da concorrência, demanda e competitividade dos cursos. Com a previsão, os estudantes podem ajustar suas

expectativas e estratégias de escolha e estudo, maximizando as chances de ingresso no curso desejado.

O modelo possibilita instituições a se prepararem de maneira mais eficaz para receber estudantes com base nas previsões de demanda, ajustando recursos e sua estrutura (docente, física e do curso) conforme necessário. Em termos de políticas públicas, governos estaduais e municipais podem utilizar essas previsões para ajustar a alocação de recursos e criar programas que incentivem a escolha, por exemplo, de universidades menos concorridas, equilibrando a distribuição de estudantes e o uso de recursos nas instituições.

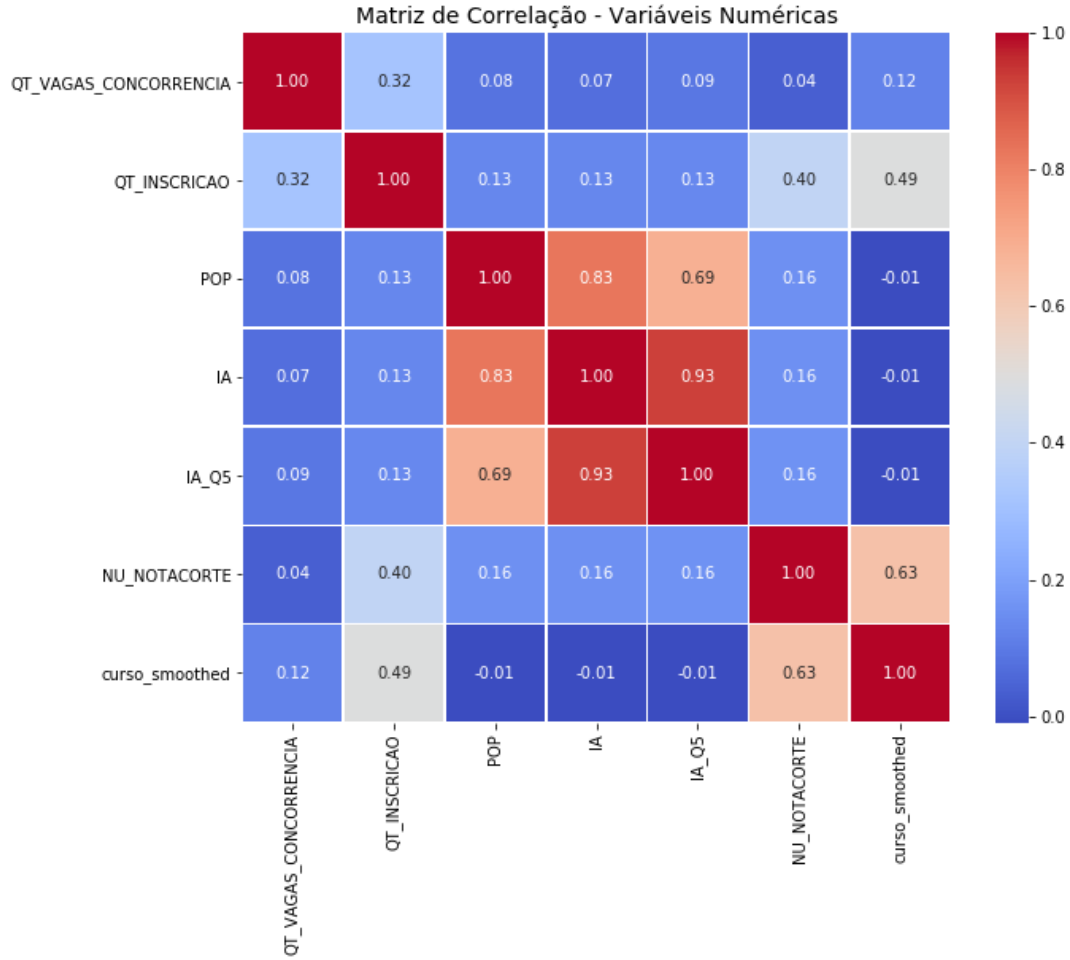
5.2. Processo de Criação dos Modelos de Machine Learning

A criação de modelos de machine learning envolve várias etapas fundamentais que guiaram o desenvolvimento do projeto. O processo iniciou com a seleção e preparação dos dados, junto a transformação e limpeza das variáveis para que pudessem ser utilizadas pelos algoritmos de aprendizado de máquina.

A seleção de variáveis foi baseada na análise exploratória, na interpretação das correlações, presentes na matriz de correlação da Figura 53, e no entendimento da importância de cada variável no contexto das notas de corte. Valores absolutos do coeficiente de correlação de Pearson maiores que 0.7 são considerados altos, logo o maior problema verificado na matriz é a alta correlação entre os índices de atratividade geral e para o ensino superior de um município. Com isso, foi preferível a utilização do índice de atração para o ensino superior, devido ao se tratar de um índice mais específico para o problema em questão.

Figura 53: Matriz de Correlação

```python
# Calcular a matriz de correlação
matriz_correlacao = data.corr()

# Gerar um heatmap para visualizar a nova correlação entre as variáveis numéricas
plt.figure(figsize=(10, 8))
sns.heatmap(matriz_correlacao, annot=True, fmt='.2f', cmap='coolwarm', cbar=True, linewidths=0.5)
plt.title('Matriz de Correlação - Variáveis Numéricas', fontsize=14)
plt.show()
```



Com isso, na Figura 54 é feita a seleção dos atributos para modelagem.

Figura 54: Seleção de Variáveis

```
sisu = pd.read_csv('sisu_trabalhado2.csv')
```

```
sisu.columns
```

```
Index(['SG_IES', 'DS_CATEGORIA_ADM', 'SG_UF_CAMPUS', 'DS_REGIAO_CAMPUS',
       'NO_MUNICIPIO_CAMPUS', 'NO_CURSO', 'DS_GRAU', 'DS_TURNO',
       'QT_VAGAS_CONCORRENCIA', 'QT_INSCRICAO', 'CI', 'IGC', 'POP',
       'HIERARQUIA_URBANA', 'COD_HIERARQUIA_URBANA', 'GRUPO_HIERARQUIA_URBANA',
       'IA', 'IA_Q5', 'NU_NOTACORTE', 'curso_smoothed'],
      dtype='object')
```

```
df = sisu[["DS_CATEGORIA_ADM","DS_REGIAO_CAMPUS","DS_GRAU","DS_TURNO","QT_VAGAS_CONCORRENCIA",
     "QT_INSCRICAO","POP","GRUPO_HIERARQUIA_URBANA","IA_Q5","CI","IGC","curso_smoothed","NU_NOTACORTE"]].copy()
```

Seguindo para o tratamento das colunas, algumas definições são necessárias. Por se tratar de notas de classificação (0, 1, 2, 3, etc..) e não de valores contínuos, os índices são definidos como *strings* para posteriormente serem tratados como categóricas ordinais. A variável de categoria administrativa possui 3 categorias nominais diferentes, mas como observado na Figura 55, apenas uma universidade é pública municipal, então é preferido dividir esta variável em dois grupos distintos apenas, sendo eles: universidades públicas federais e universidades públicas de outras categorias (neste caso, estaduais e municipais), com isso, é aplicado a codificação 0 e 1 no atributo para possibilitar a aplicação nos algoritmos.

Figura 55: Tratamento Manual dos índices e Categoria Administrativa

```
df.CI = df.CI.apply(lambda x: str(x))
df.IGC = df.IGC.apply(lambda x: str(x))
```

```
df.DS_CATEGORIA_ADM.value_counts()
```

```
Pública Federal      4468
Pública Estadual     1240
Pública Municipal       1
Name: DS_CATEGORIA_ADM, dtype: int64
```

```
df.DS_CATEGORIA_ADM = df.DS_CATEGORIA_ADM.apply(lambda x: 1 if x=="Pública Federal" else 0)
```

```
df.DS_CATEGORIA_ADM.value_counts()
```

```
1    4468
0    1241
Name: DS_CATEGORIA_ADM, dtype: int64
```

Além disso, técnicas como o ***one-hot encoding*** para variáveis categóricas, codificação de variáveis ordinais e a normalização das variáveis numéricas foram aplicadas para garantir que os dados estivessem prontos para o treinamento. Os atributos devem ser preparados para que possam ser ingeridos pelos algoritmos computacionais, muitos algoritmos apenas compreendem números e apresentam erros ao se depararem com *strings*. A normalização das variáveis numéricas é umas das técnicas mais importantes quando se trata de regressão, pois a magnitude de algumas características dos dados como população e o índice de atração são muito maiores que a das colunas como quantidade de vagas e de inscrições, podendo levar a um modelo viesado. Com a normalização, os atributos são ajustados para uma escala comum, geralmente entre 0 e 1, evitando o peso excessivo dos valores originais sobre o algoritmo de *machine learning*. As variáveis como IGC e CI são classificações ordinais, podendo ir de 0 a 5, logo o tratamento delas é diferente das categóricas nominais, como Região. Então é aplicado a função *OrdinalEncoder*, esta, diferentemente do *one-hot encoding,* não gera *dummies* para cada categoria e sim transforma os valores do atributo em inteiros, mantendo uma ordem específica definida pelo cientista de dados. Por exemplo, para o índice IGC, que possui valores de 3, 4 e 5, as categorias são atribuídas como 0, 1 e 2. Na Figura 56, todas as transformações necessárias são realizadas para então começar os treinamentos dos algoritmos de aprendizagem de máquina.

Figura 56: Transformação das Variáveis para Treinamento dos Modelos

```
# 1. Definir as colunas numéricas, categóricas e ordinais
numeric_features = ['QT_VAGAS_CONCORRENCIA','QT_INSCRICAO','POP','IA_Q5','curso_smoothed']
categorical_features = ['DS_CATEGORIA_ADM','DS_REGIAO_CAMPUS','DS_GRAU','DS_TURNO','GRUPO_HIERARQUIA_URBANA']
ordinal_features = ['CI','IGC']
```

```
df2 = df.copy()
```

```
# 2. Definir a ordem das categorias ordinais
ordinal_categories = [['2', '3', '4', '5'], ['3', '4', '5']]
```

```
# 3. Aplicar transformações separadamente

# Normalizar os dados numéricos
scaler = StandardScaler()
df2[numeric_features] = scaler.fit_transform(df2[numeric_features])

# Codificar variáveis categóricas nominais
one_hot = pd.get_dummies(df2[categorical_features])

# Codificar variáveis ordinais
ordinal = OrdinalEncoder(categories=ordinal_categories)
ord_transform = pd.DataFrame(ordinal.fit_transform(df2[ordinal_features]), columns=ordinal_features)

# 4. Combinar as transformações
df3 = pd.concat([df2[numeric_features], one_hot, ord_transform], axis=1)
```

```
df3.columns
```

```
Index(['QT_VAGAS_CONCORRENCIA', 'QT_INSCRICAO', 'POP', 'IA_Q5',
       'curso_smoothed', 'DS_CATEGORIA_ADM', 'GRUPO_HIERARQUIA_URBANA',
       'DS_REGIAO_CAMPUS_Centro_Oeste', 'DS_REGIAO_CAMPUS_Nordeste',
       'DS_REGIAO_CAMPUS_Norte', 'DS_REGIAO_CAMPUS_Sudeste',
       'DS_REGIAO_CAMPUS_Sul', 'DS_GRAU_Bacharelado', 'DS_GRAU_Licenciatura',
       'DS_GRAU_Área Básica de Ingresso (ABI)', 'DS_TURNO_Integral',
       'DS_TURNO_Matutino', 'DS_TURNO_Noturno', 'DS_TURNO_Vespertino', 'CI',
       'IGC'],
      dtype='object')
```

Com os dados preparados, o próximo passo foi a **escolha dos algoritmos de machine learning** para estudo. Foi considerado três algoritmos principais: **Support Vector Regression (SVR)**, **Regressão Ridge** e **Decision Tree Regressor**. A escolha foi influenciada pelas características dos dados e pelas necessidades do problema, especialmente considerando a presença de interações complexas e a necessidade de capturar padrões não lineares.

5.2.1. Support Vector Regression (SVR)

O **SVR** é um algoritmo derivado do *Support Vector Machines (SVM)*, porém utilizado em problemas de regressão, especialmente quando se acredita que as relações entre as variáveis não seguem padrões lineares simples. O SVR busca encontrar um hiperplano que melhor ajuste os dados, criando uma margem de tolerância ao redor da linha de regressão, dentro da qual os erros (*epsilon*) são

permitidos. Ele é ideal para modelar relações complexas, usando funções de kernel para capturar interações não lineares e ajustar os dados com base em parâmetros de regularização.

- **Vantagens**:
    - Capacidade de modelar relações complexas e não lineares. O SVR pode capturar padrões complexos, utilizando o *kernel trick*, com funções polinomiais ou radias (RBF), permitindo modelar relações não triviais entre os atributos.
    - Flexibilidade ao ajustar a margem de erro com o parâmetro *epsilon* e regularização através do parâmetro C. O algoritmo define uma margem de tolerância aos erros, permitindo lidar melhor com pequenas variações nos dados sem muita penalização e a regularização controla o *trade-off* entre o *overfitting* e o *underfitting*.
    - Robustez a valores extremos. O SVR é um algoritmo menos sensível a *outliers* em comparação com outros métodos de regressão.
- **Desvantagens**:
    - Alto custo computacional, principalmente em grandes volumes de dados. Especialmente o *kernel* RBF pode ser computacionalmente caro.
    - Sensível à escolha de hiperparâmetros como C, *epsilon* e o *kernel*, exigindo otimização criteriosa para obter os melhores resultados para melhora de seu desempenho.

5.2.2. Regressão Ridge

A **Regressão Ridge** é uma técnica de regressão linear que incorpora regularização L2 para lidar com problemas de multicolinearidade entre variáveis explicativas. A regularização atua minimizando o impacto de variáveis correlacionadas, controlando o tamanho dos coeficientes e garantindo um ajuste mais robusto, reduzindo a probabilidade de *overfitting*.

- **Vantagens**:
    - Eficiente para modelar relações lineares e lidar com multicolinearidade, reduzindo a chance de *overfitting*. O método lida muito bem como variáveis explicativas que possuem interdependência, como neste presente estudo é o caso dos índices institucionais IGC e CI.
    - Simplicidade e rapidez no treinamento, o algoritmo é computacionalmente barato em relação aos outros, além da facilidade de interpretação devido à linearidade do modelo.
- **Desvantagens**:
    - Incapacidade de capturar interações e padrões não lineares. Por ser um método puramente linear, possui dificuldades para capturar padrões mais complexos, como as interações não lineares entre competitividade do curso e demanda.
    - Menor flexibilidade em comparação a modelos como o SVR ou *Decision Tree Regressor*, limitando sua capacidade de lidar com dados mais complexos.

5.2.3. Decision Tree Regressor

O **Decision Tree Regressor** é um modelo não linear que se baseia em uma estrutura de árvore para prever os resultados. Ele divide os dados em subconjuntos com base nas variáveis explicativas, criando uma hierarquia de decisões que gera previsões numéricas a partir dos nós finais (folhas) da árvore.

- **Vantagens**:
    - Facilidade de interpretação, com regras claras de decisão. Podendo ser útil para entender quais fatores estão afetando as notas de corte.
    - Capacidade de lidar com interações não lineares e complexas. O algoritmo captura relações complexas entre as variáveis sem necessidade de muitas transformações, de maneira eficaz.

- Capacidade de lidar com valores nulos e extremos. Árvores de decisão lidam automaticamente com valores faltantes considerando folhas alternativas baseado nas informações dos dados, o que reduz a necessidade de imputações. O algoritmo também é robusto, devido ao critério de separação usado em modelos baseado em árvores ser mais resiliente a *outliers*.

- **Desvantagens**:
  - Propensão ao *overfitting*, especialmente em dados ruidosos ou quando a árvore não é adequadamente podada. O ajuste dos hiperparâmetros de forma cuidadosa é muito importante para qualidade das previsões. Além disso, árvores muito complexas e extensas podem levar ao *overfitting*.
  - Instabilidade, já que pequenas variações nos dados podem alterar drasticamente a estrutura da árvore. O método é muito sensível a estas variações.
  - Falta de continuidade. Árvores de decisão aplicadas para regressão resultam em previsões constantes por parte, devido ao particionamento do espaço de características em regiões disjuntas, logo esta limitação pode levar a previsões ruins, se tratando de uma variável predita contínua.

5.3. Treinamento e Avaliação dos Modelos de Machine Learning

5.3.1. Aplicação aos Dados

O processo de treinamento foi realizado conforme os seguintes passos:

1. **Separação dos Dados**: Os dados foram divididos em conjuntos de treino e teste com a proporção de 75% e 25%, respectivamente.

2. **Otimização dos hiperparâmetros**: Primeiramente foi realizado a otimização dos hiperparâmetros de cada modelo utilizando o método bayesiano e validação cruzada para garantir que todos os dados sejam usados no treinamento.
3. **Treinamento dos Modelos**: Com os dados já divididos em treino e teste e os hiperpâmetros otimizados já definidos, é realizado o treinamento dos modelos.
4. **Análise dos Resultados**: Os modelos foram avaliados com métricas como **Erro Absoluto Médio (EAM)**, **Erro quadrático médio (EQM)** e **R^2** para medir os erros dos modelos e a capacidade de explicação das variáveis.

5.3.2. Validação Cruzada K-Fold

Para avaliar os modelos de forma robusta, foi aplicada a técnica de **validação cruzada K-Fold**. Essa técnica divide os dados em K partes (ou folds), onde, em cada iteração, uma parte é usada como conjunto de teste e as outras K-1 partes são usadas para treinar o modelo. No estudo, foi utilizado **5 *folds***, garantindo que cada subconjunto de dados fosse utilizado tanto para treino quanto para teste, o que reduz a chance de o modelo ser otimizado apenas para um conjunto específico de dados, minimizando o risco de *overfitting*.

5.3.3. Otimização de Hiperparâmetros

Uma etapa essencial no desenvolvimento dos modelos é a **otimização dos hiperparâmetros**. Os hiperparâmetros são configurações do modelo que não são aprendidas a partir dos dados, mas que têm um impacto significativo no desempenho final. Para cada um dos três modelos, foi necessário otimizar hiperparâmetros específicos para garantir que o melhor desempenho possível fosse alcançado. Cada algoritmo possui seus próprios hiperparâmetros que controlam uma

determinada funcionalidade da aplicação e auxilia no treinamento do aprendizado de máquina.

A **Otimização Bayesiana**, implementada por meio da biblioteca **Optuna**, foi usada para realizar a busca eficiente dos hiperparâmetros. Essa técnica funciona construindo um modelo probabilístico do espaço de busca e utilizando esse modelo para selecionar os valores mais promissores de hiperparâmetros em cada iteração. Isso permite encontrar melhores configurações com menos iterações em comparação com técnicas como *Grid Search* ou *Random Search*.

Para cada um dos três modelos, a Otimização Bayesiana foi responsável por selecionar os valores ideais dos hiperparâmetros. Para melhor entendimento de cada especificidade dos algoritmos, será comentado sobre cada hiperparâmetro otimizado no estudo e seu espaço de valores definido.

5.3.3.1 Support Vector Regression (SVR)

- **C**: Controle do *trade-off* entre o erro de treinamento e a complexidade do modelo. Um valor alto de C permite menos erros no conjunto de treino, podendo levar ao *overfitting*. Já um valor baixo de C favorece um modelo mais simples, mas pode aumentar o erro de previsão, levando ao *underfitting*. O espaço de busca definido foi de **1e-6 (0,000001)** a **100**, capturando tanto modelos altamente regularizados, quanto modelos menos regularizados.

- ***epsilon***: Define a margem dentro da qual os erros não são penalizados sobre o hiperplano. Um *epsilon* menor força o modelo a ter previsões mais precisas, mas também aumenta o risco de *overfitting*. Intervalo de busca entre **0,001** e **0,5** permitindo modelos menos e mais tolerantes a erros.

- ***Kernel***: Define o tipo de função de *kernel* a ser usado. O kernel é um conjunto de funções matemáticas que os dados são submetidos para encontrar o hiperplano. Os mais utilizados são: Lineares, Não Lineares, Polinomiais e *Radial Basis Function* (RBF). Para o estudo, foi testado os tipos: **linear**, **polinomial** e **RBF**, a fim de testar os mais diferentes tipos.

5.3.3.2. Regressão Ridge

- ***Alpha***: O parâmetro alpha controla a força da regularização L2. Um valor alto de *alpha* aumenta a penalização dos coeficientes, tornando o modelo mais simples e resistente a *overfitting*, mas pode reduzir a capacidade de ajuste preciso. O espaço de busca foi de **1e-6** a **100**, possibilitando modelos com muita, média e baixa regularização.

5.3.3.3. Decision Tree Regressor

- ***max_depth***: Controla a profundidade máxima da árvore de decisão. Uma profundidade maior pode levar ao *overfitting*, pois a árvore se ajusta muito aos dados de treino. Limitar a profundidade evita que o modelo se torne muito complexo. O intervalo de valores foi de **2** a **20**, possibilitando as árvores capturarem interações importantes sem se tornar excessivamente profunda.
- ***min_samples_split***: Define o número mínimo de amostras necessárias para dividir um nó, o controle deste número garante que cada divisão ocorra apenas quando houver informações suficientes para justificar uma nova ramificação. Aumentar esse valor evita que a árvore se ramifique em ramos muito pequenos, o que pode levar ao *overfitting*. O espaço de busca foi de **2** a **20**, levando ao teste de árvores com nós muito pequenos e árvores mais suavizadas sem um ajuste excessivo.
- ***min_samples_leaf***: Define o número mínimo de amostras que cada nó folha deve conter. Isso evita que nós muito pequenos sejam criados e protege o modelo de se ajustar aos valores extremos e/ou raros. Foram utilizados valores entre **1** a **10**, possibilitando modelos com uma única amostra em cada folha ou valores maiores como 10.

As figuras abaixo evidenciam todo o processo citado acima para que fosse treinado os modelos. A Figura 60 evidencia os erros absolutos médios encontrados com os melhores parâmetros de cada processo de otimização, observa-se que o algoritmo SVR apresenta o menor valor de EAM.

Figura 57: Separação em Bases de Treino e Teste

```
# Separar os dados em features (X) e target (y)
X = df3
y = df['NU_NOTACORTE']

# Dividir os dados em treino e teste com 25% de teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Figura 58: Funções para Otimização de Hiperparâmetros de cada Modelo em análise

```
# Função objetivo para Ridge Regression com a métrica MAE
def objective_ridge(trial):
    alpha = trial.suggest_loguniform('alpha', 1e-6, 100)  # Amostrar o parâmetro alpha
    model = Ridge(alpha=alpha)

    # Avaliar o modelo com validação cruzada (k=5) usando MAE
    score = cross_val_score(model, X_train, y_train, cv=5, scoring='neg_mean_absolute_error').mean()

    # Retornar o MAE positivo (Optuna minimiza)
    return -score

# Função objetivo para Support Vector Regression (SVR) com a métrica MAE
def objective_svr(trial):
    C = trial.suggest_loguniform('C', 1e-6, 100)
    epsilon = trial.suggest_uniform('epsilon', 0.001, 0.5)
    kernel = trial.suggest_categorical('kernel', ['linear', 'poly', 'rbf'])

    model = SVR(C=C, epsilon=epsilon, kernel=kernel)

    # Avaliar o modelo com validação cruzada (k=5) usando MAE
    score = cross_val_score(model, X_train, y_train, cv=5, scoring='neg_mean_absolute_error').mean()

    # Retornar o MAE positivo (Optuna minimiza)
    return -score

# Função objetivo para Decision Tree Regressor com a métrica MAE
def objective_tree(trial):
    max_depth = trial.suggest_int('max_depth', 2, 20)
    min_samples_split = trial.suggest_int('min_samples_split', 2, 20)
    min_samples_leaf = trial.suggest_int('min_samples_leaf', 1, 10)

    model = DecisionTreeRegressor(max_depth=max_depth, min_samples_split=min_samples_split,
                                  min_samples_leaf=min_samples_leaf, random_state=42)

    # Avaliar o modelo com validação cruzada (k=5) usando MAE
    score = cross_val_score(model, X_train, y_train, cv=5, scoring='neg_mean_absolute_error').mean()

    # Retornar o MAE positivo (Optuna minimiza)
    return -score
```

Figura 59: Otimização Bayesiana dos Hiperparâmetros de cada Modelo em análise

```
# Otimizar Hiperparâmetros
# 1. Ridge Regression
study_ridge = optuna.create_study(direction='minimize')
study_ridge.optimize(objective_ridge, n_trials=50)

# 2. Support Vector Regression
study_svr = optuna.create_study(direction='minimize')
study_svr.optimize(objective_svr, n_trials=50)

# 3. Decision Tree Regressor
study_tree = optuna.create_study(direction='minimize')
study_tree.optimize(objective_tree, n_trials=50)
```

Figura 60: Hiperparâmetros otimizados de cada Modelo em análise

```
print("Best Ridge Regression parameters: ", study_ridge.best_params)
print("Best Ridge Regression MAE score: ", study_ridge.best_value)
print("Best SVR parameters: ", study_svr.best_params)
print("Best SVR MAE score: ", study_svr.best_value)
print("Best Decision Tree parameters: ", study_tree.best_params)
print("Best Decision Tree MAE score: ", study_tree.best_value)
```

```
Best Ridge Regression parameters:  {'alpha': 1.0038785486465014e-06}
Best Ridge Regression MAE score:  36.34199812457766
Best SVR parameters:  {'C': 99.79862226659922, 'epsilon': 0.07957601995992755, 'kernel': 'rbf'}
Best SVR MAE score:  28.101311453818518
Best Decision Tree parameters:  {'max_depth': 17, 'min_samples_split': 17, 'min_samples_leaf': 3}
Best Decision Tree MAE score:  29.400207879417543
```

Com os melhores hiperparâmetros, a etapa de treinamento foi realizada, Figura 61. Finalizando a etapa de treino dos modelos de aprendizado de máquina e analisando a Tabela 5, identificou-se que o melhor algoritmo para esta problemática é o SVR. Os menores valores de erro foram encontrados (EAM de 28,28 e EQM de 1475,67) e para o coeficiente de determinação (R^2) foi encontrado o maior valor dentre os modelos (0,71).

Tabela 5: Métricas dos Modelos em análise

ALGORITMO	EAM	EQM	R^2
REGRESSÃO RIDGE	35,44	2153,21	0,57
***SUPPORT VECTOR REGRESSION* (SVR)**	**28,28**	**1473,67**	**0,71**
DECISION TREE REGRESSOR	29,50	1499,26	0,70

Figura 61: Treinamento dos Modelos em análise

```
# Avaliar os modelos finais com os melhores parâmetros
# Avaliar Ridge Regression
ridge_best = Ridge(**study_ridge.best_params)
ridge_best.fit(X_train, y_train)
ridge_pred = ridge_best.predict(X_test)
ridge_results = {
    'mae': mean_absolute_error(y_test, ridge_pred),
    'mse': mean_squared_error(y_test, ridge_pred),
    'r2': r2_score(y_test, ridge_pred)
}

# Avaliar SVR
svr_best = SVR(**study_svr.best_params)
svr_best.fit(X_train, y_train)
svr_pred = svr_best.predict(X_test)
svr_results = {
    'mae': mean_absolute_error(y_test, svr_pred),
    'mse': mean_squared_error(y_test, svr_pred),
    'r2': r2_score(y_test, svr_pred)
}

# Avaliar Decision Tree Regressor
tree_best = DecisionTreeRegressor(**study_tree.best_params)
tree_best.fit(X_train, y_train)
tree_pred = tree_best.predict(X_test)
tree_results = {
    'mae': mean_absolute_error(y_test, tree_pred),
    'mse': mean_squared_error(y_test, tree_pred),
    'r2': r2_score(y_test, tree_pred)
}

# Imprimir os resultados finais
print("Ridge Regression Results:", ridge_results)
print("SVR Results:", svr_results)
print("Decision Tree Results:", tree_results)
```

```
Ridge Regression Results: {'mae': 35.34425321569969, 'mse': 2153.2153263621885, 'r2': 0.5733014968786018}
SVR Results: {'mae': 28.2802837018753, 'mse': 1475.668493281776, 'r2': 0.7075696380767689}
Decision Tree Results: {'mae': 29.504505845679933, 'mse': 1499.2459725858396, 'r2': 0.7028973347528757}
```

5.4. Avaliação do Modelo SVR

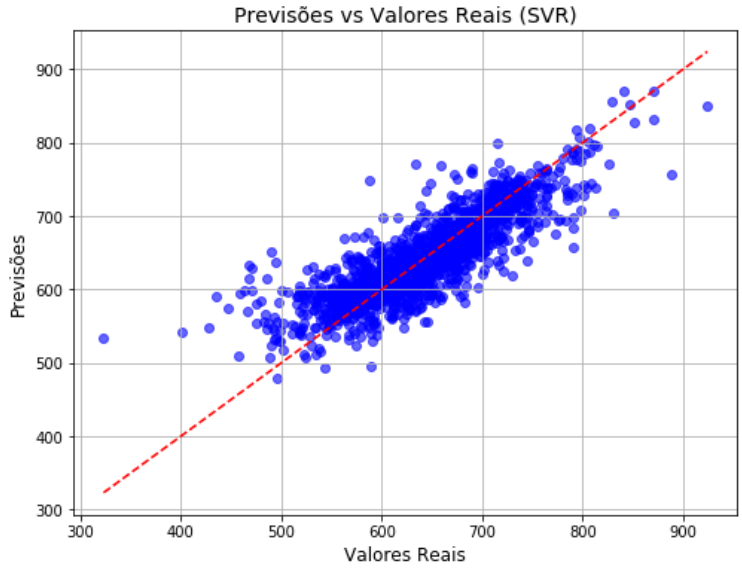
Após a otimização dos hiperparâmetros e treinamento dos modelos, o algoritmo do Support Vector Regression (SVR) foi selecionado como o melhor modelo. Próximo passo foi avaliá-lo utilizando gráficos e métricas de desempenho que ajudam a entender a qualidade de suas previsões e identificar possíveis melhorias.

O gráfico de dispersão, presente na Figura 62, mostra a relação entre as previsões do modelo e os valores reais dos dados de teste. No eixo X estão os valores reais e, no eixo Y, as previsões feitas pelo SVR. A linha de perfeição (y = x), representada em vermelho, indica onde as previsões seriam exatamente iguais aos valores reais, simbolizando o modelo “perfeito “.

Observa-se que a maioria dos pontos estão relativamente próximos da linha de perfeição, indicando que o modelo está fazendo boas previsões para os dados. Isso sugere que o modelo conseguiu capturar bem as relações presentes nos dados de treino, no entanto, alguns pontos estão mais distantes da linha, o que indica que há previsões com erros maiores, evidenciando que aprimoramentos no ajuste do modelo possam ser benéficos em futuros estudos.

Figura 62: Gráfico de Dispersão – Previsões vs Valores Reais

```
# Gráfico de dispersão (scatter plot) das previsões vs valores reais
plt.figure(figsize=(8, 6))
plt.scatter(y_test, svr_pred, alpha=0.6, color='b')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')  # Linha de perfeição (y=x)
plt.title('Previsões vs Valores Reais (SVR)', fontsize=14)
plt.xlabel('Valores Reais', fontsize=12)
plt.ylabel('Previsões', fontsize=12)
plt.grid(True)
plt.show()
```



Outra forma gráfica de avaliar o modelo é com um histograma dos resíduos, o qual apresenta a distribuição dos erros de previsão do modelo, chamados de resíduos. Um resíduo é definido como a diferença entre o valor real e a previsão realizada pelo modelo;

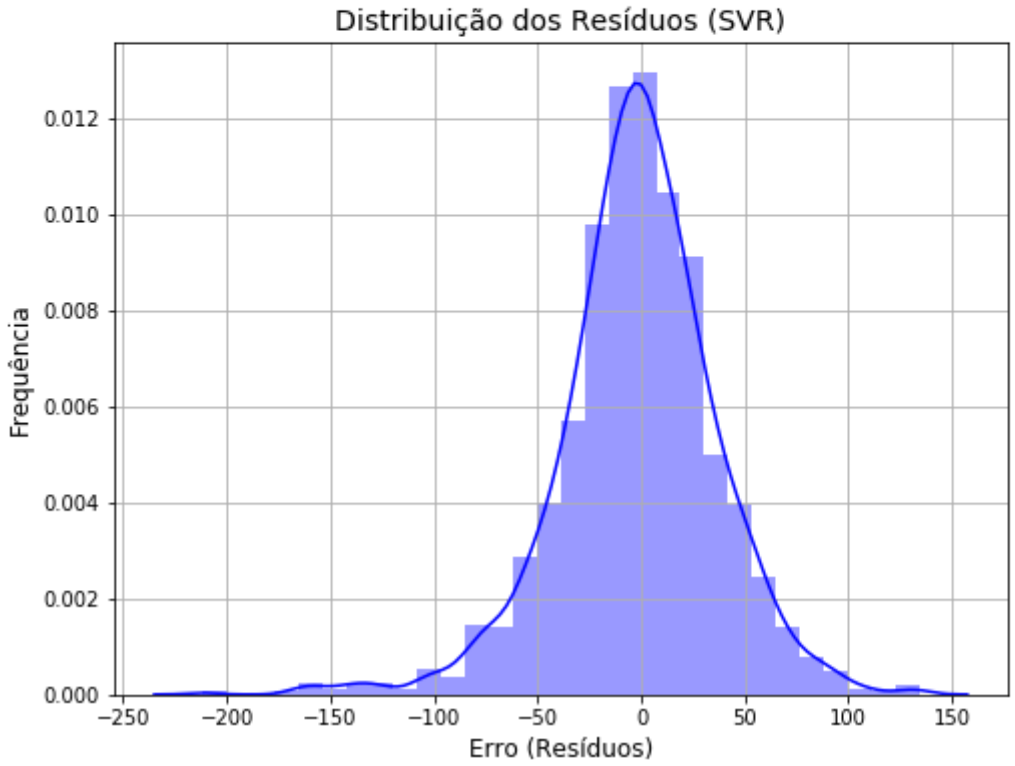
Resíduo = Valor Real - Previsão.

Para um modelo bem ajustado, o que se espera da distribuição ideal são resíduos simetricamente distribuídos em torno de zero. Isso indicaria que o modelo não está tendencioso e que está cometendo erros tanto para cima quanto para baixo de forma equilibrada.

Pela Figura 63, o que o gráfico mostra que os resíduos estão concentrados ao redor de zero, com uma distribuição relativamente simétrica. Isso sugere que o modelo não possui um viés sistemático para superestimar ou subestimar os valores, o que é um indicativo de que as previsões estão, em média, bem balanceadas. Nota-se que alguns poucos cursos estão sendo previstos com notas bem acima dos reais cortes, indicando um leve viés para um lado, porém a ausência de grandes outliers é um bom sinal, mostrando que o modelo não está cometendo grandes erros para a grande maioria das observações.

Figura 63: Histograma dos Resíduos

```python
# Histograma dos resíduos (erros de previsão)
residuos = y_test - svr_pred
plt.figure(figsize=(8, 6))
sns.distplot(residuos, kde=True, color='blue', bins=30)
plt.title('Distribuição dos Resíduos (SVR)', fontsize=14)
plt.xlabel('Erro (Resíduos)', fontsize=12)
plt.ylabel('Frequência', fontsize=12)
plt.grid(True)
plt.show()
```



De maneira geral, a análise gráfica indica que o modelo SVR obteve um bom desempenho em capturar a maioria dos padrões e relações dos dados, fazendo previsões razoavelmente precisas, com erros bem distribuídos**.**

Por último é avaliado as métricas do modelo, presentes na Tabela 5, o coeficiente de determinação (R^2) é uma métrica que indica a proporção da variação na variável dependente (notas de corte) que é explicada pelo modelo. Para o SVR, o valor de **R^2** foi de **0,71**, ou seja, 71% da variabilidade nas notas de corte foi capaz de ser explicada pelo modelo com base nas variáveis preditoras. Isso indica que o modelo está capturando bem as relações entre as variáveis e as notas de corte, fornecendo previsões que estão em boa concordância com os valores reais.

Além do coeficiente de determinação, foram utilizadas duas métricas importantes para avaliar a precisão das previsões do modelo final: o **Erro Quadrático Médio (EQM)** e o **Erro Absoluto Médio (EAM)**. O valor do EQM foi de **1475,67**, o que significa que, em média, o quadrado da diferença entre os valores reais e as previsões foi de 1475,67. O EQM penaliza de forma mais significativa grandes erros de previsão, pois o erro é elevado a segunda potência. Este valor sugere que, embora o modelo faça boas previsões em geral, ainda existem algumas observações onde os erros são mais pronunciados, foi possível visualizar estes erros maiores no histograma dos resíduos.

Já o erro absoluto médio foi de **28,28**, o que significa que, em média, o modelo SVR errou as previsões das notas de corte em aproximadamente 28,28 pontos. O EAM mede a magnitude média dos erros sem considerar o seu sinal, e neste caso, indica que o modelo erra, em média, a cerca de 28 pontos de distância dos reais cortes dos cursos. Isso sugere que, para a maioria dos cursos, o modelo SVR está fornecendo previsões que estão próximas das notas de corte reais, o que pode ser útil para orientar estudantes e instituições em suas decisões.

Essas métricas, junto ao R^2 de 71% e a análise gráfica, indicam que o modelo SVR é eficaz na previsão das notas de corte, com erros relativamente pequenos em termos absolutos, mas futuros trabalhos podem refinar a análise para reduzir os grandes erros observados em alguns casos específicos e melhorar ainda mais a compreensão das complexas relações entre as variáveis.

6. Apresentação dos Resultados

O objetivo principal deste presente estudo foi prever as notas de corte dos cursos, mais especificamente da modalidade de ampla concorrência presenciais de licenciatura e bacharelado cadastrados no sistema do MEC para que com isso, candidatos e instituições possam obter valiosas informações para auxiliá-los em tomadas de decisão.

Com base nos resultados obtidos e na análise dos algoritmos de machine learning, o modelo SVR se destacou como a melhor opção para prever as notas de

corte dos cursos do SISU, graças à sua capacidade de capturar padrões complexos e não lineares. A utilização de técnicas de regularização e a otimização dos hiperparâmetros permitiram um bom ajuste, com erros pequenos e boa capacidade de generalização. O uso do *kernel* RBF permitiu ao modelo capturar relações não lineares complexas entre as variáveis explicativas e as notas de corte, essencial para entender interações entre demanda dos cursos, qualidade institucional e contexto regional, que afetam de maneira não trivial as notas de corte. O parâmetro **C**, que controla o trade-off entre a margem de erro e a complexidade do modelo, foi ajustado para um valor relativamente alto (**99,8**), permitindo ao modelo focar em previsões precisas pela avaliação do modelo, o alto valor pareceu não comprometer a generalização, não sendo percebido um *overfitting* do modelo. O **epsilon** de 0,079 mostrou que o modelo permitiu pequenas variações nos dados sem a penalização excessiva dos erros, lidando com outliers e ruídos nos dados, fornecendo previsões mais estáveis, possivelmente balanceando o valor de controle utilizado.

Além disso, com a função *permutation_importance* foi calculada a **importância das variáveis** nas previsões. A técnica de ***Permutation Importance*** revela quais variáveis tiveram o maior impacto nas previsões, pela Figura 65 e 66, essas incluíram o curso aplicado *Target Encoding,* a quantidade de inscrições e quantidade de vagas disponíveis como os três atributos mais significantes para a previsão, sugerindo que o prestígio, a demanda e a competitividade pelo curso foram fatores cruciais na definição das notas de corte. Logo após, os fatores demográficos como índice de atração pelo ensino superior e a população do município se mostram importantes também. OS fatores institucionais IGC e CI também obtiveram uma importância razoável nas previsões, isso oferece insights valiosos para o planejamento educacional e para os candidatos no processo de escolha dos cursos.

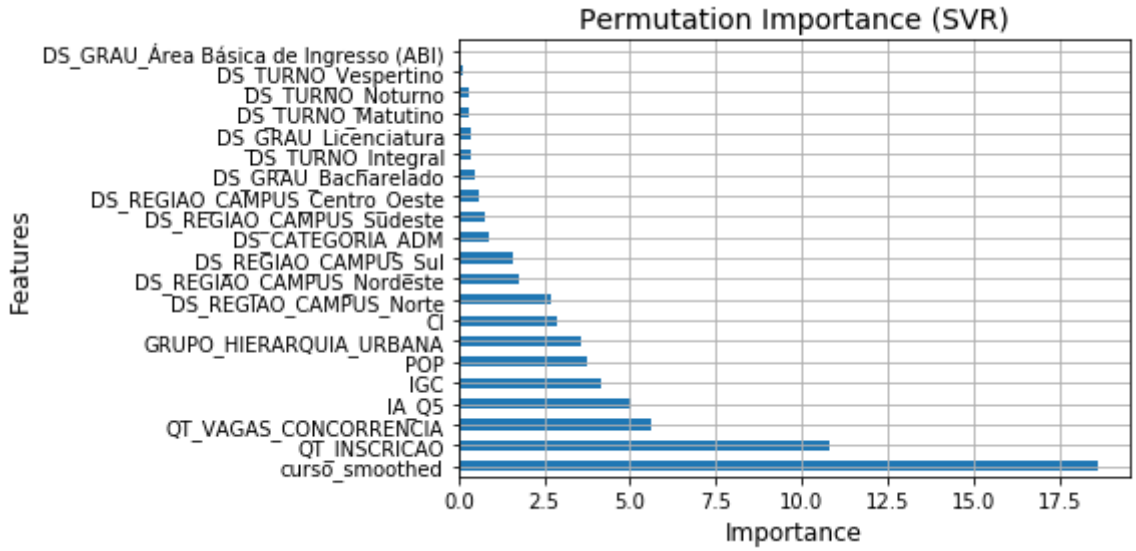
Figura 64: Script para geração de gráfico da importância das variáveis do modelo

```python
# Avaliar a importância das features usando Permutation Importance
result = permutation_importance(svr_best, X_test, y_test, n_repeats=10, random_state=42, scoring='neg_mean_absolute_error')

# Exibir a importância das features
importance_df = pd.DataFrame(result.importances_mean, index=X.columns, columns=['Importance'])\
                  .sort_values(by='Importance', ascending=False)
print(importance_df)

# Gráfico da importância das variáveis
plt.figure(figsize=(10, 6))
importance_df.plot(kind='barh', legend=False)
plt.title('Permutation Importance (SVR)', fontsize=14)
plt.xlabel('Importance', fontsize=12)
plt.ylabel('Features', fontsize=12)
plt.grid(True)
plt.show()
```

Figura 65: Valores da importância das variáveis do modelo

	Importance
curso_smoothed	18.611339
QT_INSCRICAO	10.829508
QT_VAGAS_CONCORRENCIA	5.623530
IA_Q5	5.046452
IGC	4.164389
POP	3.732275
GRUPO_HIERARQUIA_URBANA	3.548805
CI	2.874119
DS_REGIAO_CAMPUS_Norte	2.693966
DS_REGIAO_CAMPUS_Nordeste	1.735216
DS_REGIAO_CAMPUS_Sul	1.603329
DS_CATEGORIA_ADM	0.877201
DS_REGIAO_CAMPUS_Sudeste	0.770375
DS_REGIAO_CAMPUS_Centro_Oeste	0.607541
DS_GRAU_Bacharelado	0.463188
DS_TURNO_Integral	0.375810
DS_GRAU_Licenciatura	0.365659
DS_TURNO_Matutino	0.316434
DS_TURNO_Noturno	0.293310
DS_TURNO_Vespertino	0.122428
DS_GRAU_Área Básica de Ingresso (ABI)	0.014873

Figura 66: Gráfico de barras da importância da variáveis do modelo



Os insights obtidos com o modelo podem ajudar tanto candidatos quanto instituições de ensino. Para os candidatos, as previsões fornecem uma estimativa realista das notas de corte, permitindo que se preparem melhor para o processo seletivo. Para as instituições e gestores, os resultados podem auxiliar na otimização da alocação de vagas e na formulação de políticas educacionais mais eficientes.

Por fim, apesar dos bons resultados, ainda há espaço para melhorias, futuros estudos poderiam trabalhar na inclusão de novas variáveis e dados, como: inclusão de dados de anos anteriores, adição de informações socioeconômicas dos municípios, métricas de qualidade dos cursos como (CPC e ENADE) e características como número de professores doutorados e índice de evasão dos cursos podem auxiliar na previsão das notas. A experimentação com outros modelos mais complexos, pode refinar ainda mais as previsões e oferecer um panorama mais completo do comportamento das notas de corte no SISU. A utilização de métodos de *ensemble* e o uso de redes neurais profundas (*Deep Learning*) podem capturar de forma mais eficaz os padrões nos dados e prover melhores resultados.

7. Links

Link para o vídeo: https://youtu.be/2GlDnCAfAWY

Link para o repositório: https://github.com/MateusFelipes/TCC_POS_PUCMINAS