MBA Admission dataset, Class 2025

Descrição do banco de dados:

1. Data Source: Synthetic data generated from the Wharton Class of 2025's statistics.

2. Meta Data: application_id: Unique identifier for each application gender: Applicant's gender (Male, Female) international: International student (TRUE/FALSE) gpa: Grade Point Average of the applicant (on 4.0 scale) major: Undergraduate major (Business, STEM, Humanities) race: Racial background of the applicant (e.g., White, Black, Asian, Hispanic, Other / null: international student) gmat: GMAT score of the applicant (800 points) work_exp: Number of years of work experience (Year) work_industry: Industry of the applicant's previous work experience (e.g., Consulting, Finance, Technology, etc.) admission: Admission status (Admit, Waitlist, Null: Deny)

Objetivos

Qual a distribuição de gênero dos candidatos e como isso se relaciona com o status de admissão (Admit, Waitlist, Deny)?

Estudantes internacionais têm maiores ou menores chances de serem admitidos em comparação com estudantes nacionais?

Como o GPA influencia o status de admissão? Existe uma correlação clara entre o GPA e a aceitação?

Quais majors (Business, STEM, Humanities) têm as maiores taxas de admissão? Algum campo se destaca?

A pontuação do GMAT dos candidatos afeta significativamente a chance de admissão? Existe um limite de pontuação mais frequente entre os admitidos?

Como a experiência de trabalho (anos de experiência e indústria) impacta o status de admissão? Setores específicos, como Consultoria ou Tecnologia, estão mais associados a admissões?

Existe alguma correlação entre a raça dos candidatos (para não internacionais) e as taxas de admissão?

Importação de pacotes

In [74]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from statsmodels.stats.proportion import proportions_ztest
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTEN
from collections import Counter
```

Data Wrangling: Preparação e Limpeza dos Dados

Agora vamos explorar o conjunto de dados para identificar possíveis inconsistências, dados ausentes e realizar as transformações necessárias antes de aplicarmos nossos modelos de machine learning. A etapa de Data Wrangling é crucial, pois garante que o dataset esteja devidamente estruturado e pronto para ser analisado. Além disso, permite detectar problemas como valores duplicados, variáveis irrelevantes ou inconsistentes, que podem prejudicar a performance do modelo.

Durante essa fase, revisamos colunas, tipos de dados e padrões faltantes, além de realizar ajustes nas variáveis categóricas e numéricas para garantir que estejam adequadas para a modelagem preditiva.

Esse processo técnico é fundamental para assegurar a qualidade do nosso modelo e a integridade dos resultados.

In [75]:
```
dados = pd.read_csv('MBA.csv')
dados.head()
```

Out[75]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_industr
0	1	Female	False	3.30	Business	Asian	620.0	3.0	Financia Service
1	2	Male	False	3.28	Humanities	Black	680.0	5.0	Investmen Managemen
2	3	Female	True	3.30	Business	NaN	710.0	5.0	Technolog
3	4	Male	False	3.47	STEM	Black	690.0	6.0	Technolog
4	5	Male	False	3.35	STEM	Hispanic	590.0	5.0	Consultin

In [76]:
```
dados.isna().sum()
```

Out[76]:
```
application_id       0
gender               0
international        0
gpa                  0
major                0
race              1842
gmat                 0
work_exp             0
work_industry        0
admission         5194
dtype: int64
```

In [77]:
```
dados['admission'].unique()
```

Out[77]:
```
array(['Admit', nan, 'Waitlist'], dtype=object)
```

Obs:

Na coluna Admissão, encontramos duas categorias: "Admit" para os alunos que foram aceitos no programa e "Waitlist" para aqueles que ficaram na lista de espera. No entanto, os alunos que não foram aceitos estão representados por valores ausentes (NaN). Para garantir a consistência do dataset, vamos tratar esses valores ausentes, assumindo que todos os NaN representam alunos que não foram aceitos no programa.

Vamos substituir os valores NaN por "Not Accepted", criando assim uma terceira categoria explícita e facilitando a análise posterior.

In [78]:
```
dados['admission'].fillna('NotAccepted', inplace=True)
dados['admission'].unique()
```

Out[78]:
```
array(['Admit', 'NotAccepted', 'Waitlist'], dtype=object)
```

In [79]:
```
dados.isna().sum()
```

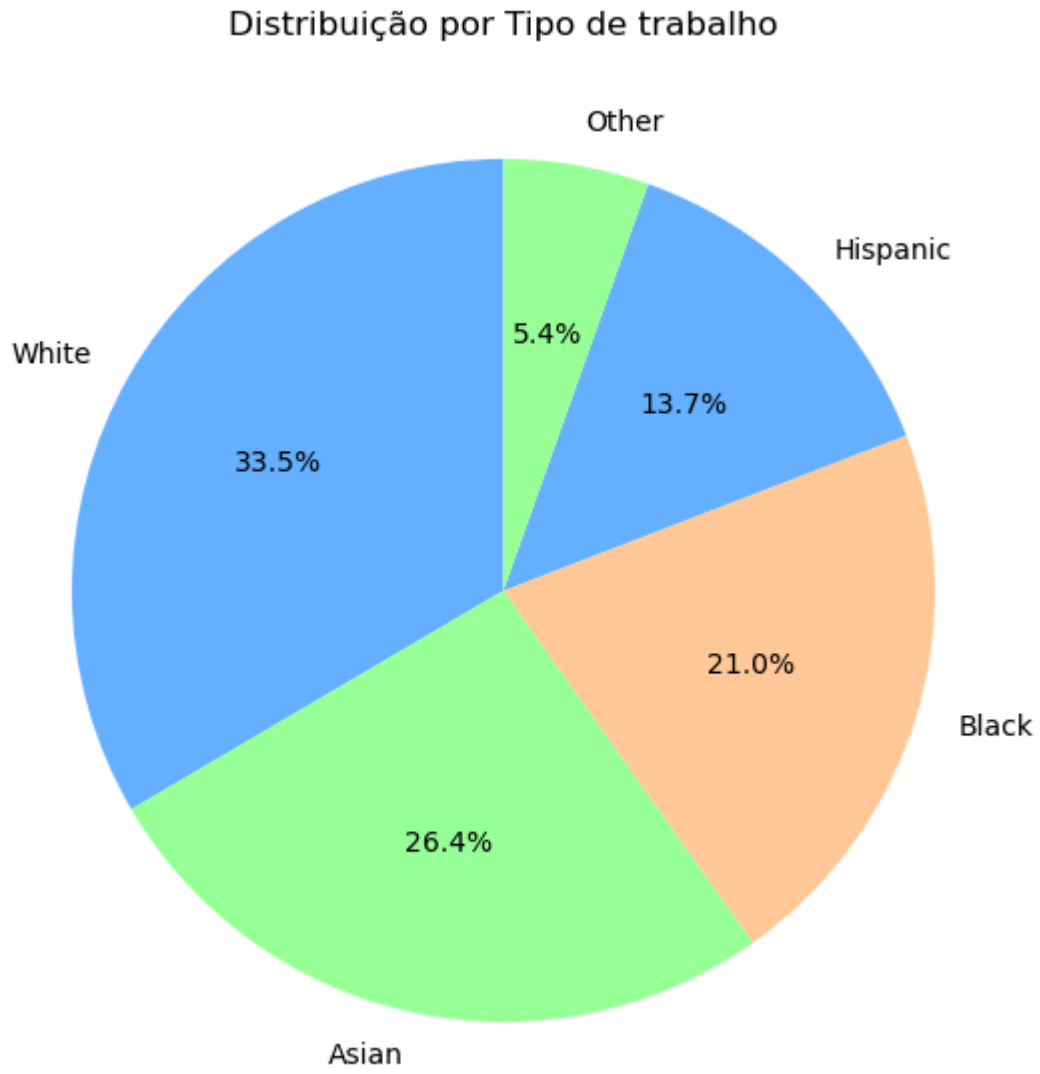
Out[79]:
```
application_id       0
gender               0
international        0
gpa                  0
major                0
race              1842
gmat                 0
work_exp             0
work_industry        0
admission            0
dtype: int64
```

In [80]:
```
dados['race'].unique()
```

Out[80]:
```
array(['Asian', 'Black', nan, 'Hispanic', 'White', 'Other'], dtype=object)
```

In [81]:
```
tipo_trabalho = dados['race'].value_counts()

# Criar o gráfico de pizza
plt.figure(figsize=(7,7))
plt.pie(tipo_trabalho, labels=tipo_trabalho.index, autopct='%1.1f%%', startangle=90
plt.title('Distribuição por Tipo de trabalho')
plt.show()
```



In [82]:
```
dados['race'].isna().sum() / dados['race'].shape[0]
```

Out[82]:
```
0.2973845657087504
```

In [83]:
```
dados['race'].fillna('NaoEspecificado', inplace=True)
dados['race'].unique()
```

Out[83]:
```
array(['Asian', 'Black', 'NaoEspecificado', 'Hispanic', 'White', 'Other'],
      dtype=object)
```

Obs:

Como os dados são categóricos e alguns valores estão ausentes, optamos por criar uma nova categoria chamada NAoEspecificado. Essa abordagem permite que não percamos observações, 29,73% dos dados, valiosas ao excluir linhas ou colunas, além de evitar suposições que poderiam distorcer os resultados ao preencher os valores com uma média ou mediana. A nova categoria servirá para indicar que os dados daquela variável específica não foram fornecidos ou não se aplicam, mantendo a integridade da análise e permitindo a interpretação adequada dessas observações.

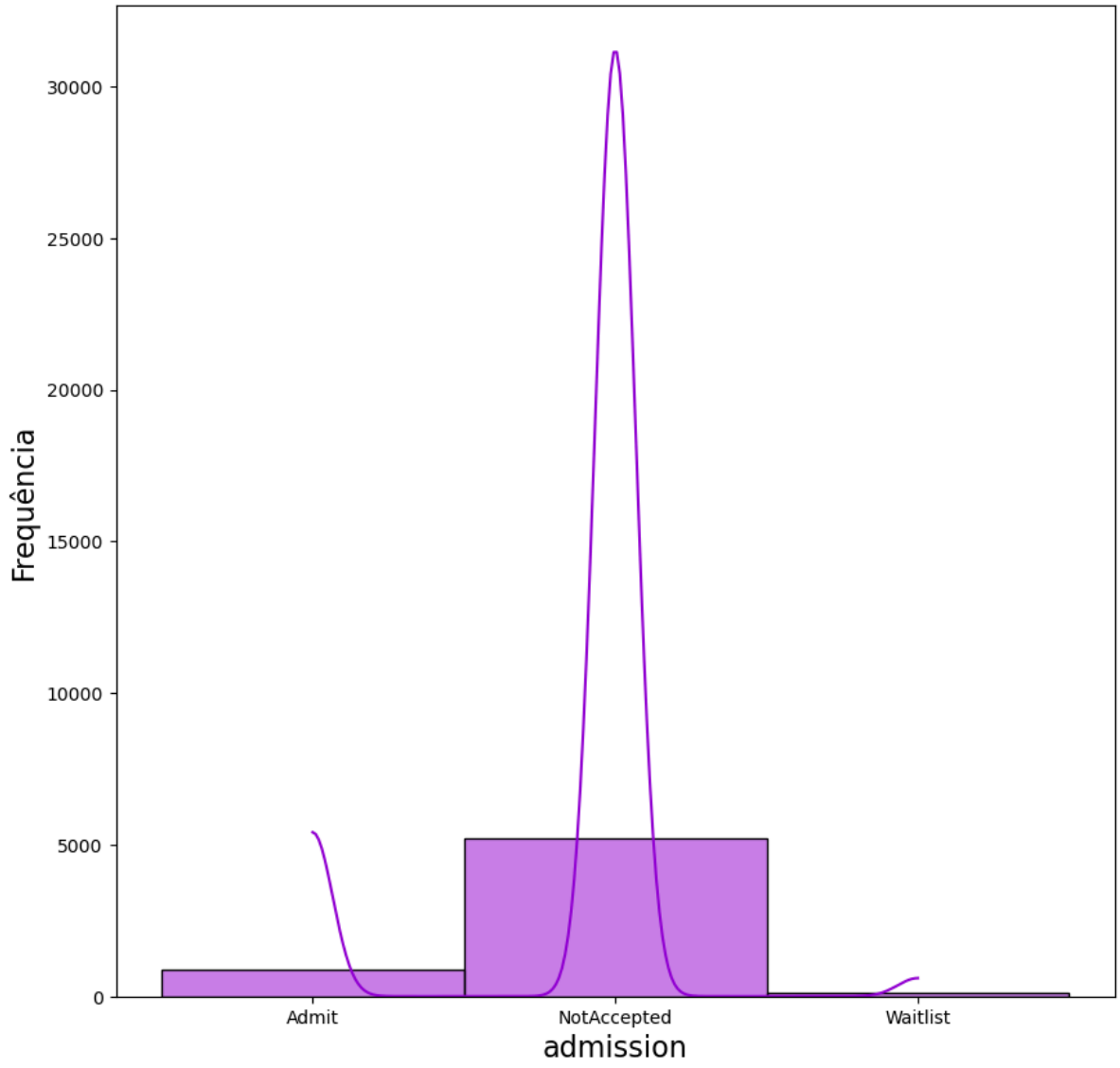
In [84]:
```
dados.isna().sum()
```

Out[84]:
```
application_id    0
gender            0
international     0
gpa               0
major             0
race              0
gmat              0
work_exp          0
work_industry     0
admission         0
dtype: int64
```

EDA

In [85]:
```
def hist(df, col):
    plt.figure(figsize=(10,10))
    sns.histplot(data = df[col], kde=True, bins=30, color='darkviolet')
    plt.xlabel(col, fontsize=16)
    plt.ylabel('Frequência', fontsize=16)
    plt.show()
```

In [86]:
```
hist(dados,'admission')
```



In [87]:
```
adimitidos = dados[dados['admission'] == 'Admit'].shape[0]
NaoAdimitidos = dados[dados['admission'] == 'NotAccepted'].shape[0]
alunos = dados.shape[0]
padmitidos = (adimitidos/alunos)*100
```

```
pNaoAdimitidos = (NaoAdimitidos/alunos)*100

print(f'Quantidade de candidatos aceitos: {adimitidos}, propocional de {padmitidos:
print(f'Quantidade de candidatos aceitos: {NaoAdimitidos}, propocional de {pNaoAdim
```

```
Quantidade de candidatos aceitos: 900, propocional de 14.53%, ou seja, 15 a cada 1
00 são aceitos
Quantidade de candidatos aceitos: 5194, propocional de 83.86%, ou seja, 84 a cada
100 não são aceitos
```

Obs:

Nesse momento, identificamos que enfrentaremos dificuldades em prever a variável target, cujo objetivo é determinar se o aluno foi admitido, uma vez que as categorias estão desbalanceadas. Quando há uma grande disparidade entre as classes, como muitos casos de "não admitido" e poucos de "admitido", os modelos de aprendizado de máquina tendem a se inclinar para a classe majoritária, comprometendo a precisão das previsões.

Isso nos leva à necessidade de ajustar a abordagem, utilizando técnicas como re-amostragem (oversampling ou undersampling), pesos ajustados para classes no algoritmo ou até mesmo considerando métricas específicas para avaliar o desempenho em dados desbalanceados, como a AUC-ROC ou F1-score, que lidam melhor com esse tipo de cenário.

1 - Qual a distribuição de gênero dos candidatos e como isso se relaciona com o status de admissão (Admit, Waitlist, Deny)?

In [88]:
```
dados['gender'].unique()
```

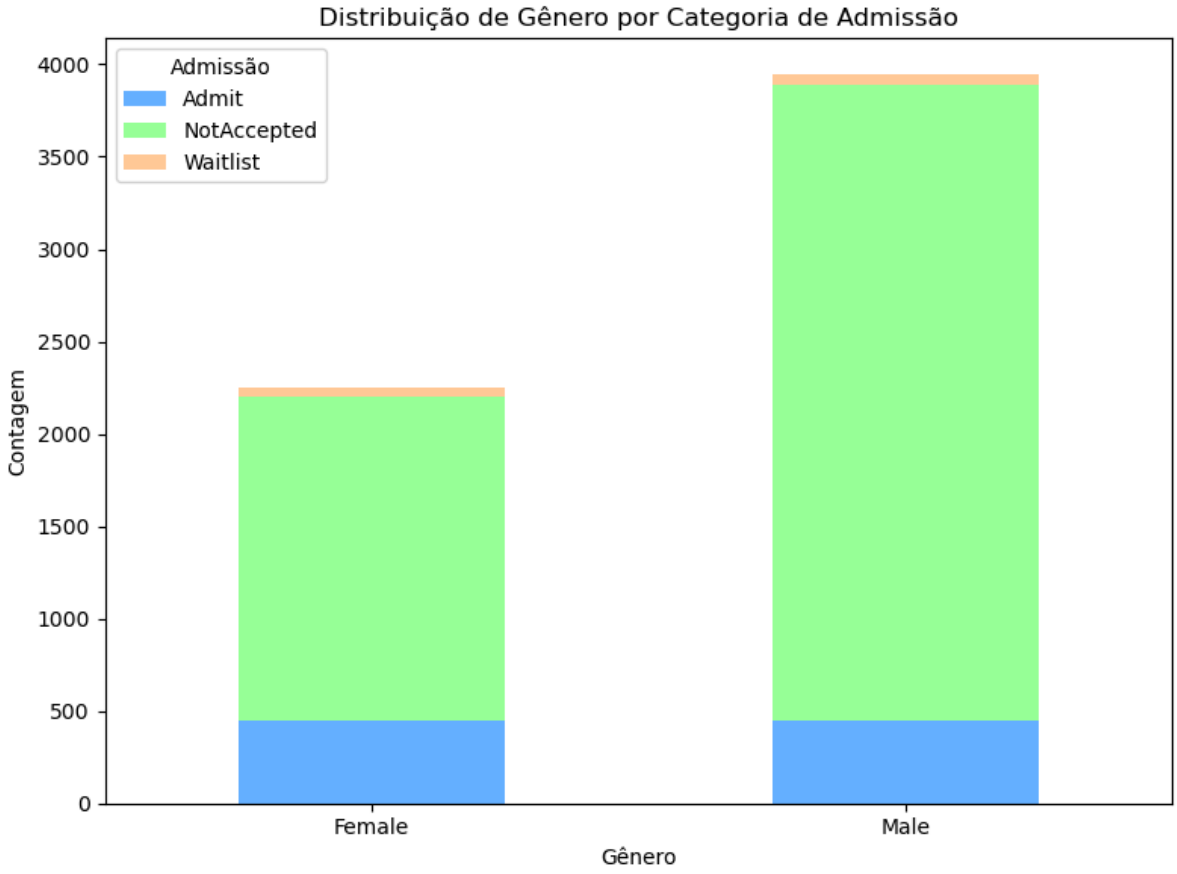
Out[88]:
```
array(['Female', 'Male'], dtype=object)
```

In [89]:
```
# Contar as frequências por Gênero e Categoria de Admissão
admissao_genero = dados.groupby(['gender', 'admission']).size().unstack(fill_value=

# Criar o gráfico de barras empilhadas
admissao_genero.plot(kind='bar', stacked=True, color=['#66b3ff', '#99ff99', '#ffcc9

# Título e rótulos
plt.title('Distribuição de Gênero por Categoria de Admissão')
plt.xlabel('Gênero',)
plt.ylabel('Contagem')
plt.xticks(rotation=0)


# Exibir o gráfico
plt.legend(title='Admissão')
plt.tight_layout()
plt.show()
```



Obs:

A análise sugere que, ao considerarmos os números absolutos, ambos os gêneros têm o mesmo número de aprovados. No entanto, quando observamos a proporção, as mulheres demonstram uma vantagem em relação à categoria de admissão. Para validar essa observação, podemos realizar um teste de proporções utilizando a estatística Z.

Passos para Realizar o Teste de Proporções Definir os Grupos: Precisamos identificar quantas pessoas de cada gênero foram admitidas e quantas não foram admitidas.

Calcular as Proporções: Utilizar os dados para calcular as proporções de aprovados entre homens e mulheres.

Realizar o Teste Z: Usar o teste Z de proporções para verificar se a diferença observada é estatisticamente significativa.

In [90]:
```
homem = dados[(dados['gender'] == 'Male')].shape[0]
mulher = dados[(dados['gender'] == 'Female')].shape[0]
mulherAdmitida = dados[(dados['gender'] == 'Female') & (dados['admission'] == 'Admi
homemAdmitido = dados[(dados['gender'] == 'Male') & (dados['admission'] == 'Admit')
ph = (homemAdmitido/homem) * 100
pm = (mulherAdmitida/mulher) * 100


print(f'Quantidade de candidatos aceitos do sexo masculino: {homemAdmitido}, propoc
print(f'Quantidade de candidatos aceitos do sexo feminino: {mulherAdmitida}, propoc
```

```
Quantidade de candidatos aceitos do sexo masculino: 450, propocional de 11.41%, ou
seja, 11 a cada 100 são aceitos
Quantidade de candidatos aceitos do sexo feminino: 450, propocional de 19.99%, ou
seja, 20 a cada 100 não são aceitos
```

In [91]:
```
def teste_proporcao(count1, n1, count2, n2):
    # Realiza o teste Z de proporções
    count = np.array([count1, count2])
    nobs = np.array([n1, n2])

    # Teste Z
    stat, pval = proportions_ztest(count, nobs)

    print(f'Estatística Z: {stat:.4f}')
    print(f'Valor-p: {pval:.4f}')

    # Interpretação
    if pval < 0.05:
        print("Há uma diferença significativa entre as proporções.")
    else:
        print("Não há diferença significativa entre as proporções.")


# Chama a função para realizar o teste Z
teste_proporcao(homemAdmitido, homem, mulherAdmitida, mulher)
```

```
Estatística Z: -9.2148
Valor-p: 0.0000
Há uma diferença significativa entre as proporções.
```

Resposta 1:

Estatística Z: -9.2148: Esse valor negativo e muito alto em magnitude sugere que a proporção de aprovação entre os gêneros é substancialmente diferente, com uma grande discrepância em relação ao que seria esperado se não houvesse diferença.

Valor-p: 0.0000: Um valor-p tão baixo (inferior a 0.05) indica que a probabilidade de observar uma diferença tão extrema entre as proporções de aprovação dos gêneros, assumindo que não há diferença real, é praticamente zero. Isso nos leva a rejeitar a hipótese nula.

A análise sugere que há uma diferença significativa entre as proporções de aprovação dos gêneros. Embora ambos os gêneros tenham o mesmo número absoluto de aprovados, a proporção revela que as mulheres estão em uma posição mais favorável quando se trata de admissão. Esse resultado é importante e pode ter implicações relevantes para políticas de admissão, incentivando uma análise mais profunda sobre as razões por trás dessas disparidades.

2 - Estudantes internacionais têm maiores ou menores chances de serem admitidos em comparação com estudantes nacionais?

Obs:

Para analisar a diferença entre alunos do país e alunos de fora do país em relação à admissão, seguimos a mesma abordagem utilizada na comparação anterior entre gêneros. Vamos começar fazendo uma breve descrição do que vamos avaliar e depois apresentar os resultados do teste de proporções.

In [92]:
```
dados['international'].unique()
```

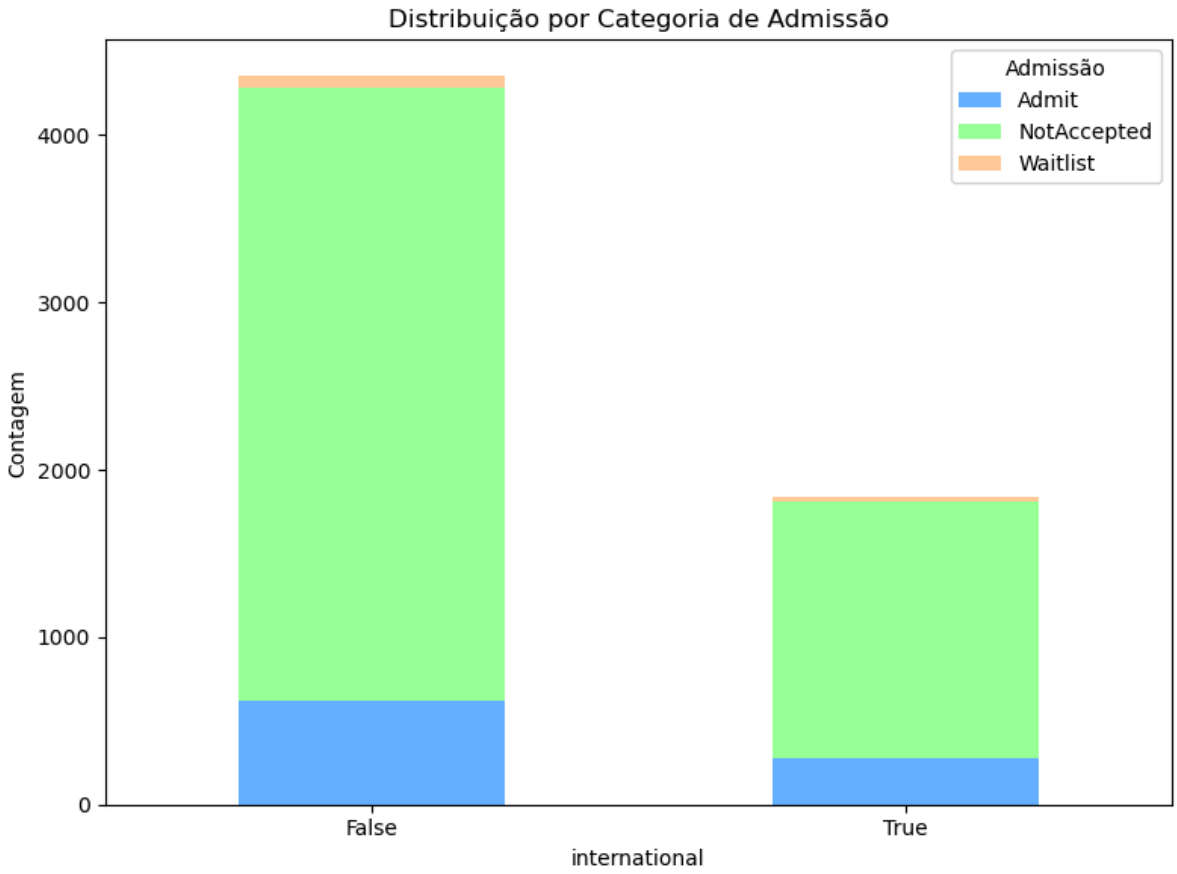
Out[92]:
```
array([False,  True])
```

In [93]:
```
# Contar as frequências por Gênero e Categoria de Admissão
admissao_genero = dados.groupby(['international', 'admission']).size().unstack(fill

# Criar o gráfico de barras empilhadas
admissao_genero.plot(kind='bar', stacked=True, color=['#66b3ff', '#99ff99', '#ffcc9

# Título e rótulos
plt.title('Distribuição por Categoria de Admissão')
plt.xlabel('international',)
plt.ylabel('Contagem')
plt.xticks(rotation=0)


# Exibir o gráfico
plt.legend(title='Admissão')
plt.tight_layout()
plt.show()
```



In [94]:
```
internacional = dados[(dados['international'] == True)].shape[0]
nacional = dados[(dados['international'] == False)].shape[0]
nacionalAdmitida = dados[(dados['international'] == False) & (dados['admission'] ==
internacionalAdmitido = dados[(dados['international'] == True) & (dados['admission'
pi = (internacionalAdmitido/internacional) * 100
pn = (nacionalAdmitida/nacional) * 100


print(f'Quantidade de candidatos aceitos de países estrangeiros: {internacionalAdmi
print(f'Quantidade de candidatos aceitos do país sede: {nacionalAdmitida}, propocio
```

```
Quantidade de candidatos aceitos de países estrangeiros: 278, propocional de 15.0
9%, ou seja, 15 a cada 100 são aceitos
Quantidade de candidatos aceitos do país sede: 622, propocional de 14.29%, ou sej
a, 14 a cada 100 não são aceitos
```

In [95]:
```
# Chama a função para realizar o teste Z
teste_proporcao(internacionalAdmitido, internacional, nacionalAdmitida, nacional)
```

```
Estatística Z: 0.8167
Valor-p: 0.4141
Não há diferença significativa entre as proporções.
```

Resposta 2:

Na nossa análise, avaliamos se existe uma diferença significativa nas taxas de admissão entre alunos nacionais e internacionais. Os resultados obtidos foram os seguintes:

Estatística Z: 0.8167 Valor-p: 0.4141 Interpretação dos Resultados Estatística Z: Um valor de 0.8167 sugere que a diferença entre as proporções observadas (admitidos de cada grupo) não é suficientemente grande para indicar uma discrepância significativa em relação à hipótese nula.

Valor-p: Com um valor-p de 0.4141, que é bem maior que 0.05, não há evidência estatística para rejeitar a hipótese nula. Isso indica que não existe uma diferença significativa nas proporções de admissão entre alunos nacionais e internacionais.

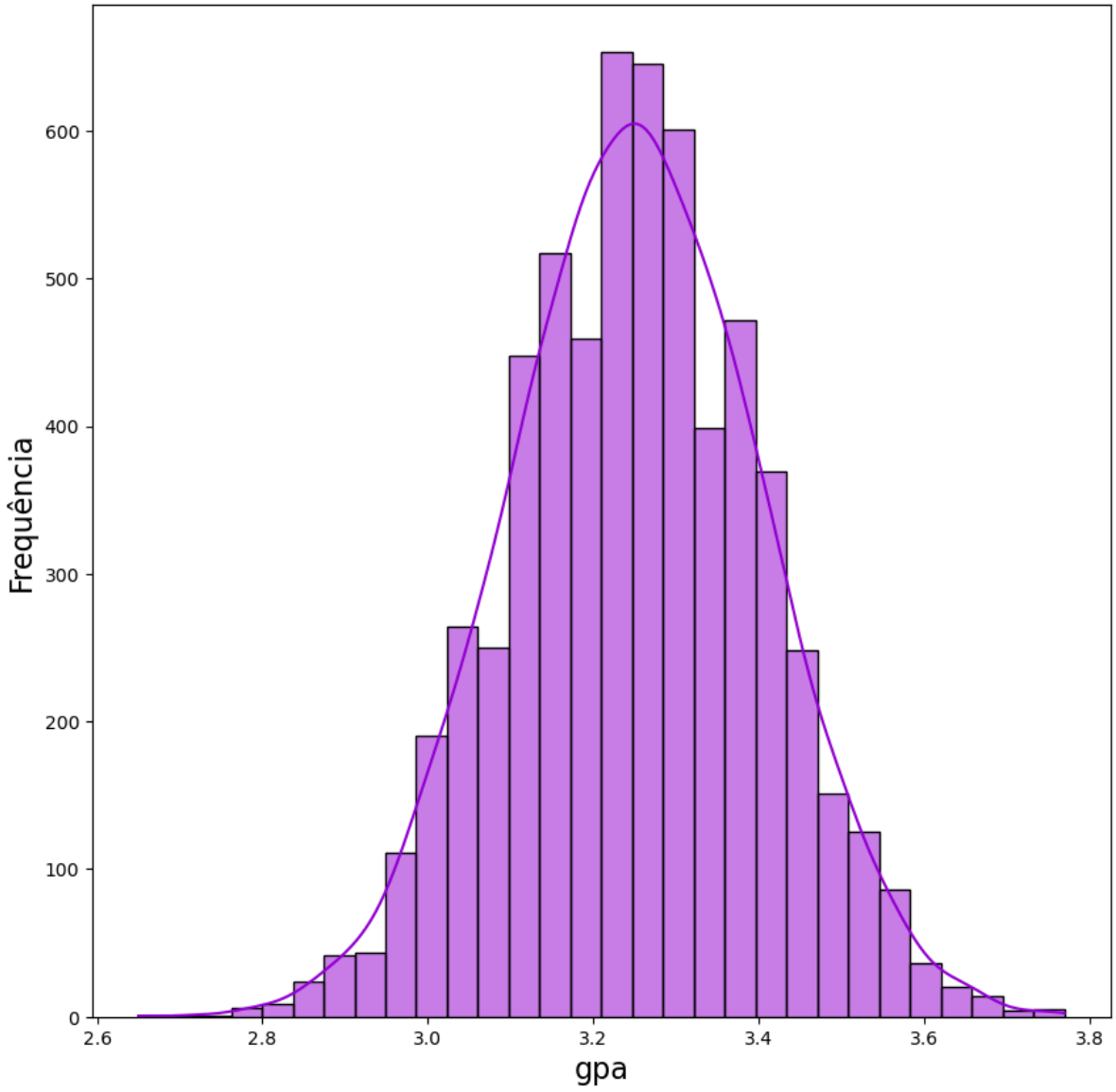
Conclusão Com base nos resultados, podemos concluir que a origem dos alunos (nacional ou internacional) não tem um impacto significativo nas taxas de admissão. Isso sugere que as políticas de admissão são igualmente eficazes para ambos os grupos ou que outros fatores não analisados podem estar influenciando as decisões de admissão.

3 - Como o GPA influencia o status de admissão? Existe uma correlação clara entre o GPA e a aceitação?

In [96]:
```
dados['gpa'].describe()
```

Out[96]:
```
count    6194.000000
mean        3.250714
std         0.151541
min         2.650000
25%         3.150000
50%         3.250000
75%         3.350000
max         3.770000
Name: gpa, dtype: float64
```

In [97]:
```
hist(dados,'gpa')
```



In [98]:
```
# Mapear valores
gpa = dados.copy()
gpa.head()
```

Out[98]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_
0	1	Female	False	3.30	Business	Asian	620.0	3.0	
1	2	Male	False	3.28	Humanities	Black	680.0	5.0	In Man
2	3	Female	True	3.30	Business	NaoEspecificado	710.0	5.0	Te
3	4	Male	False	3.47	STEM	Black	690.0	6.0	Te
4	5	Male	False	3.35	STEM	Hispanic	590.0	5.0	C

In [99]:
```
mapping = {
    'NotAccepted': 0,
    'Waitlist': 1,
    'Admit': 2
}
```

```
gpa['admission_Numeric'] = gpa['admission'].map(mapping)
gpa.head()
```

Out[99]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_
0	1	Female	False	3.30	Business	Asian	620.0	3.0	
1	2	Male	False	3.28	Humanities	Black	680.0	5.0	In Man
2	3	Female	True	3.30	Business	NaoEspecificado	710.0	5.0	Te
3	4	Male	False	3.47	STEM	Black	690.0	6.0	Te
4	5	Male	False	3.35	STEM	Hispanic	590.0	5.0	C

In [100…

```
correlacao = gpa['admission_Numeric'].corr(gpa['gpa'])

print(f'A correlação entre GPA e Admissão é: {correlacao:.2f}')
```

```
A correlação entre GPA e Admissão é: 0.29
```

Resposta 3:

Um coeficiente de correlação de 0.29 indica uma correlação positiva moderada entre GPA e a probabilidade de admissão. Isso sugere que, à medida que o GPA dos candidatos aumenta, há uma tendência de que suas chances de serem admitidos também aumentem, embora essa relação não seja extremamente forte. Essa correlação sugere que o GPA pode ser um dos fatores considerados no processo de admissão, mas não é o único determinante. Outros fatores também podem desempenhar papéis importantes nas decisões de admissão.

A correlação não implica causalidade. Portanto, não podemos afirmar que um GPA mais alto causa uma maior taxa de admissão. É importante investigar se existem outras variáveis que possam estar influenciando a admissão.

4 - Quais majors (Business, STEM, Humanities) têm as maiores taxas de admissão? Algum campo se destaca?

In [101…

```
dados['major'].unique()
```

Out[101]:

```
array(['Business', 'Humanities', 'STEM'], dtype=object)
```

In [102…

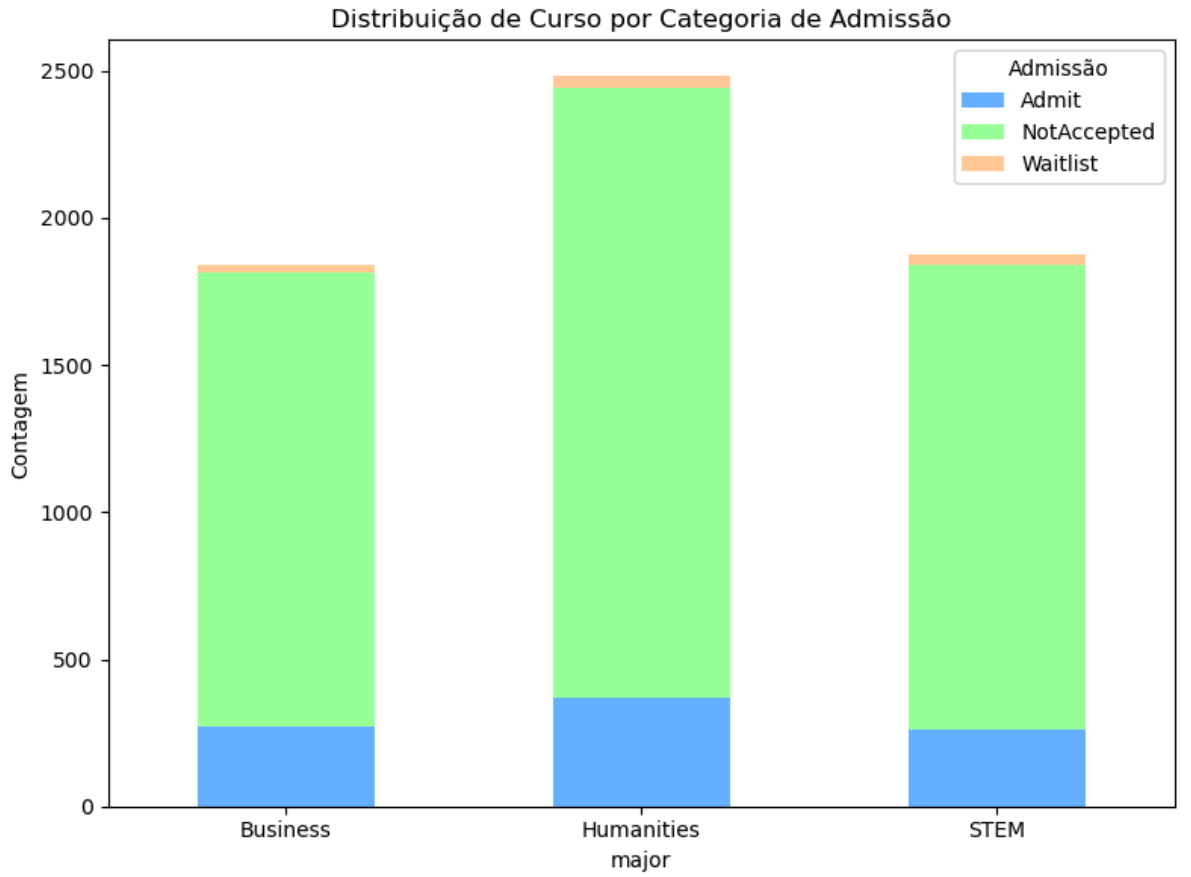
```
# Contar as frequências por Gênero e Categoria de Admissão
admissao_genero = dados.groupby(['major', 'admission']).size().unstack(fill_value=0

# Criar o gráfico de barras empilhadas
admissao_genero.plot(kind='bar', stacked=True, color=['#66b3ff', '#99ff99', '#ffcc9

# Título e rótulos
plt.title('Distribuição de Curso por Categoria de Admissão')
plt.xlabel('major',)
plt.ylabel('Contagem')
plt.xticks(rotation=0)


# Exibir o gráfico
plt.legend(title='Admissão')
```

```
plt.tight_layout()
plt.show()
```



In [103…
```
dados.head()
```

Out[103]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_
0	1	Female	False	3.30	Business	Asian	620.0	3.0	
1	2	Male	False	3.28	Humanities	Black	680.0	5.0	In Man
2	3	Female	True	3.30	Business	NaoEspecificado	710.0	5.0	Te
3	4	Male	False	3.47	STEM	Black	690.0	6.0	Te
4	5	Male	False	3.35	STEM	Hispanic	590.0	5.0	C

Obs:

A comparação do número de aprovados por grupo de curso é uma questão complexa que vai além da simples contagem. Aqui estão alguns pontos importantes a serem considerados:

1. Vagas Ofertadas vs. Aprovados 1.1. Vagas Disponíveis: Diferentes cursos têm diferentes números de vagas disponíveis. Um curso que não preenche todas as suas vagas pode, à primeira vista, parecer menos competitivo, mas isso não necessariamente reflete a qualidade ou a dificuldade do processo de admissão.

   1.2. Dificuldade Relativa: Para realmente avaliar a dificuldade de admissão em cada curso, precisamos considerar não apenas o número de aprovados, mas também quantas vagas foram oferecidas. Por exemplo, um curso com poucas vagas e muitos

candidatos pode ser mais difícil de entrar do que um curso com muitas vagas e menos candidatos.

2. Proporcionalidade Proporção de Aprovados: Avaliar a taxa de aprovação (número de aprovados dividido pelo número de inscritos) para cada curso pode nos dar uma visão melhor da competitividade. Essa análise permitirá observar como cada grupo de curso se comporta em relação ao número de candidatos e às vagas disponíveis.

3. Intersecção de Candidatos

   3.1. Candidatos Repetidos: Outro fator a considerar é a possibilidade de candidatos que se inscrevem em mais de um curso. A intersecção de candidatos pode levar a uma análise mais complexa, já que alguns candidatos podem ter sido aprovados em mais de um curso. Isso complicaria a contagem simples de aprovados por curso.

   3.2. Escolha de Cursos Secundários: Além disso, se os candidatos têm a opção de escolher um curso secundário para tentar a aprovação, isso também deve ser considerado. A capacidade de se inscrever em múltiplos cursos pode influenciar a taxa de aprovação e a análise geral.

Portanto, a análise da aprovação por grupo de curso deve ser abordada com cautela, levando em conta não apenas o número absoluto de aprovados, mas também a proporção em relação ao número de vagas disponíveis, a intersecção de candidatos e a possibilidade de escolha de cursos secundários. Uma análise mais robusta e cuidadosa garantirá que as conclusões tiradas sejam justas e representativas do cenário real.

In [104…
```
dados.major.unique()
```

Out[104]:
```
array(['Business', 'Humanities', 'STEM'], dtype=object)
```

In [105…
```
Business = dados[(dados['major'] == 'Business')].shape[0]
Humanities = dados[(dados['major'] == 'Humanities')].shape[0]
STEM = dados[(dados['major'] == 'STEM')].shape[0]
BusinessAdmitida = dados[(dados['major'] == 'Business') & (dados['admission'] == 'A
HumanitiesAdmitido = dados[(dados['major'] == 'Humanities') & (dados['admission'] =
STEMAdmitido = dados[(dados['major'] == 'STEM') & (dados['admission'] == 'Admit')].
pb = (BusinessAdmitida/Business) * 100
ph = (HumanitiesAdmitido/Humanities) * 100
ps = (STEMAdmitido/STEM) * 100


print(f'Quantidade de candidatos aceitos de Business: {BusinessAdmitida} de {Busine
print(f'Quantidade de candidatos aceitos de Humanities: {HumanitiesAdmitido} de {Hu
print(f'Quantidade de candidatos aceitos de STEM: {STEMAdmitido} de {STEM} candidat
```

```
Quantidade de candidatos aceitos de Business: 270 de 1838 candidatos, propocional
de 14.69%, ou seja, 15 a cada 100 são aceitos
Quantidade de candidatos aceitos de Humanities: 367 de 2481 candidatos, propociona
l de 14.79%, ou seja, 15 a cada 100 não são aceitos
Quantidade de candidatos aceitos de STEM: 263 de 1875 candidatos, propocional de 1
4.03%, ou seja, 14 a cada 100 não são aceitos
```

Resposta 4:

As taxas de aceitação para Business (14,69%) e Humanities (14,79%) são muito próximas, sugerindo que esses cursos têm uma concorrência semelhante. A taxa de aceitação para

STEM (14,03%) é ligeiramente inferior, o que pode indicar uma competição mais acirrada nesse campo ou uma maior seletividade por parte dos avaliadores. Diante dos dados ficaram algumas lacunas para que possamos avaliar melhor quanto a concorrência deles.

Esses dados revelam um panorama interessante sobre a competitividade entre os cursos. A taxa de aceitação pode ajudar futuros candidatos a entenderem melhor suas chances de admissão em cada área e a formularem estratégias adequadas para melhorar suas candidaturas. Além disso, pode ser útil para as instituições ajustarem suas práticas de admissão, garantindo que a seleção seja justa e equilibrada para todos os grupos de candidatos.

5 - A pontuação do GMAT dos candidatos afeta significativamente a chance de admissão? Existe um limite de pontuação mais frequente entre os admitidos?

In [106…
```
dados['gmat'].describe()
```

Out[106]:
```
count    6194.000000
mean      651.092993
std        49.294883
min       570.000000
25%       610.000000
50%       650.000000
75%       680.000000
max       780.000000
Name: gmat, dtype: float64
```

In [107…
```
dados.head()
```

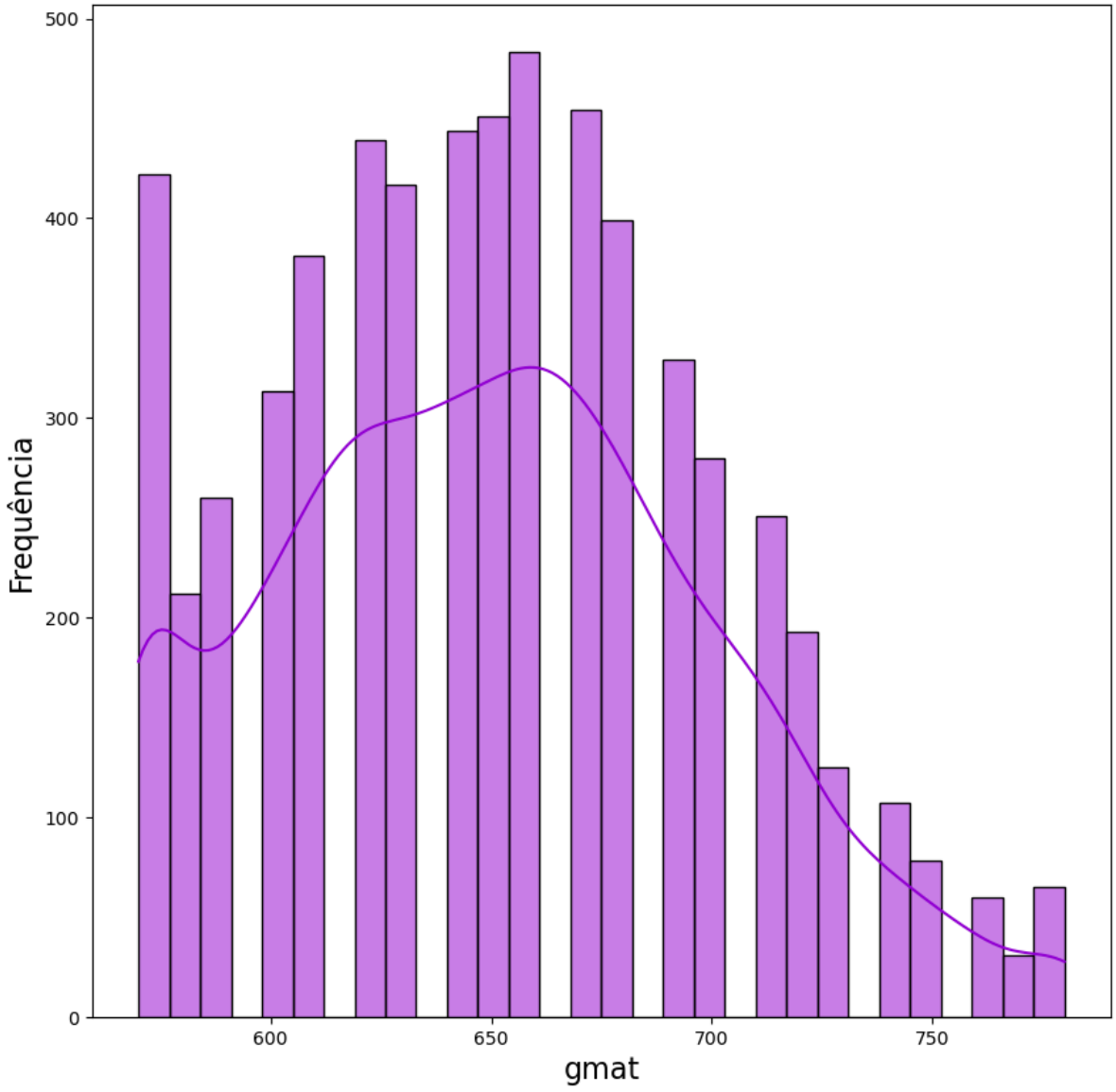
Out[107]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_
0	1	Female	False	3.30	Business	Asian	620.0	3.0	
1	2	Male	False	3.28	Humanities	Black	680.0	5.0	In Man
2	3	Female	True	3.30	Business	NaoEspecificado	710.0	5.0	Te
3	4	Male	False	3.47	STEM	Black	690.0	6.0	Te
4	5	Male	False	3.35	STEM	Hispanic	590.0	5.0	C

Obs:

Nesta análise, vamos investigar a relação entre as pontuações do GMAT dos candidatos e suas taxas de admissão no programa. Essa relação é importante, pois pode indicar se a pontuação do GMAT é um fator significativo no processo de seleção.

In [108…
```
hist(dados,'gmat')
```



In [109…

```
correlacao = gpa['admission_Numeric'].corr(gpa['gmat'])

print(f'A correlação entre GMAT e Admissão é: {correlacao:.2f}')
```

```
A correlação entre GMAT e Admissão é: 0.36
```

Resposta 5:

Uma correlação de 0.36 sugere que, à medida que a pontuação do GMAT aumenta, a probabilidade de admissão também tende a aumentar. Isso significa que candidatos com melhores desempenhos no GMAT têm uma chance maior de serem admitidos. Pode ser interpretada como um indicativo de que o GMAT tem alguma relevância como critério de avaliação, mas não é o único fator determinante na decisão de admissão. Isso sugere que, embora a pontuação no GMAT seja importante, outros critérios, como GPA, cartas de recomendação, experiência profissional e entrevistas, também desempenham papéis significativos.

6 - Como a experiência de trabalho (anos de experiência e indústria) impacta o status de admissão? Setores específicos, como Consultoria ou Tecnologia, estão mais associados a admissões?

Obs:

Nesta primeira análise, vamos examinar a relação entre o número de anos de experiência dos candidatos e suas chances de admissão no programa. A ideia é verificar se há uma correlação significativa entre essas duas variáveis, o que pode indicar que candidatos com mais experiência têm maior probabilidade de serem aceitos.

Na segunda análise, vamos explorar a relação entre a indústria dos candidatos e suas taxas de admissão, que nos ajudará a identificar padrões de aceitação entre diferentes indústrias.

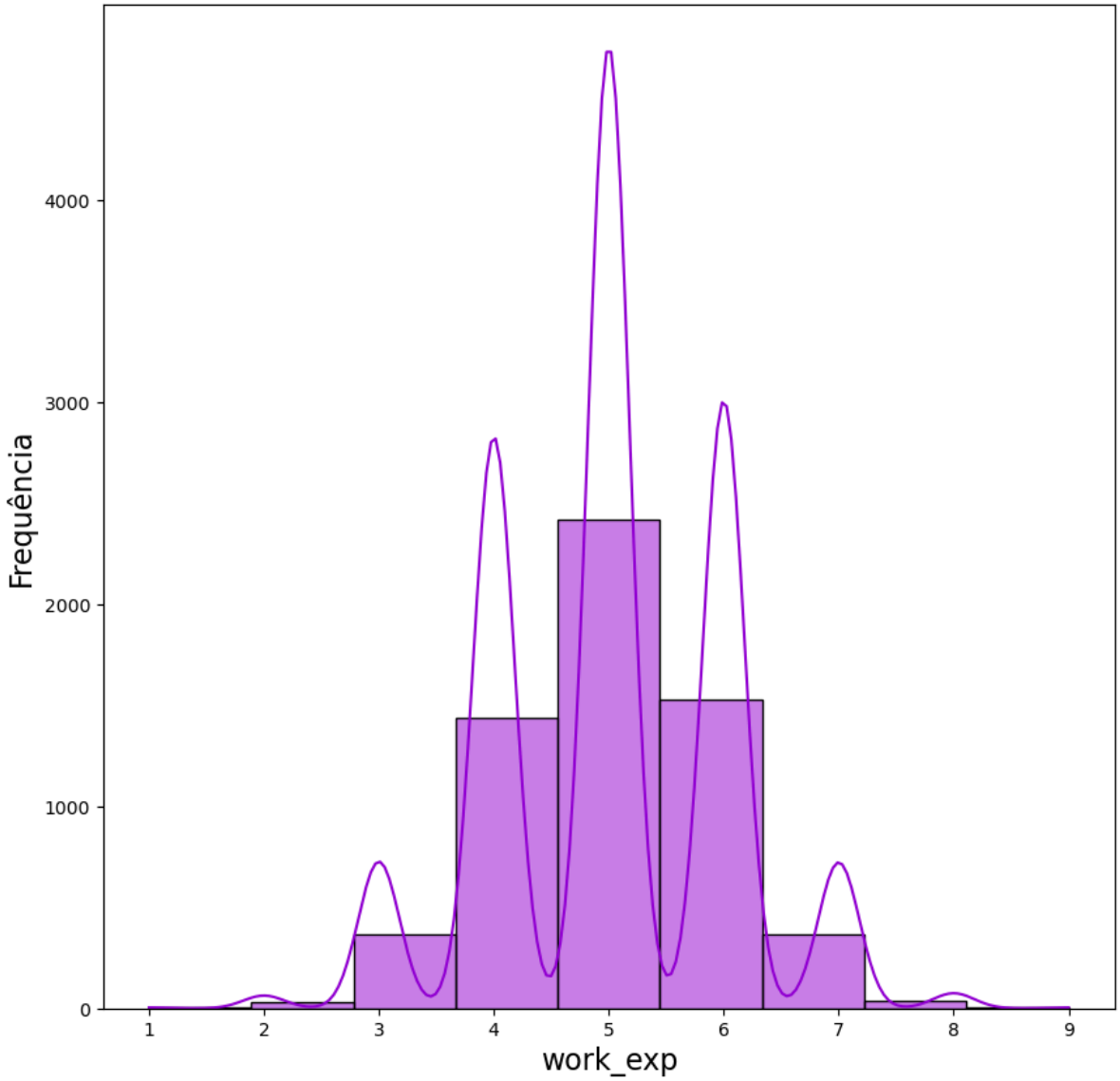
Essas duas análises complementares nos proporcionarão uma visão mais abrangente sobre como diferentes fatores, como a experiência e a indústria, influenciam as decisões de admissão. Ao final, seremos capazes de tirar conclusões informadas que poderão orientar futuras decisões sobre o processo de admissão.

In [110…
```
dados.head()
```

Out[110]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_
0	1	Female	False	3.30	Business	Asian	620.0	3.0	
1	2	Male	False	3.28	Humanities	Black	680.0	5.0	In Man
2	3	Female	True	3.30	Business	NaoEspecificado	710.0	5.0	Te
3	4	Male	False	3.47	STEM	Black	690.0	6.0	Te
4	5	Male	False	3.35	STEM	Hispanic	590.0	5.0	C

In [111…
```
def hist(df, col):
    plt.figure(figsize=(10,10))
    sns.histplot(data = df[col], kde=True, bins=9, color='darkviolet')
    plt.xlabel(col, fontsize=16)
    plt.ylabel('Frequência', fontsize=16)
    plt.show()
hist(dados,'work_exp')
```



In [112…
```
correlacao = gpa['admission_Numeric'].corr(gpa['work_exp'])

print(f'A correlação entre Anos de Experiência e Admissão é: {correlacao:.2f}')
```

```
A correlação entre Anos de Experiência e Admissão é: 0.01
```

Obs:

O valor de correlação de 0.01 entre anos de experiência e admissão sugere que essa variável não é um fator determinante no processo de seleção. Isso abre a oportunidade de investigar mais a fundo outros aspectos que podem influenciar a admissão e de ajustar as estratégias de candidatura dos candidatos para maximizar suas chances.

In [113…
```
dados['work_industry'].unique()
```

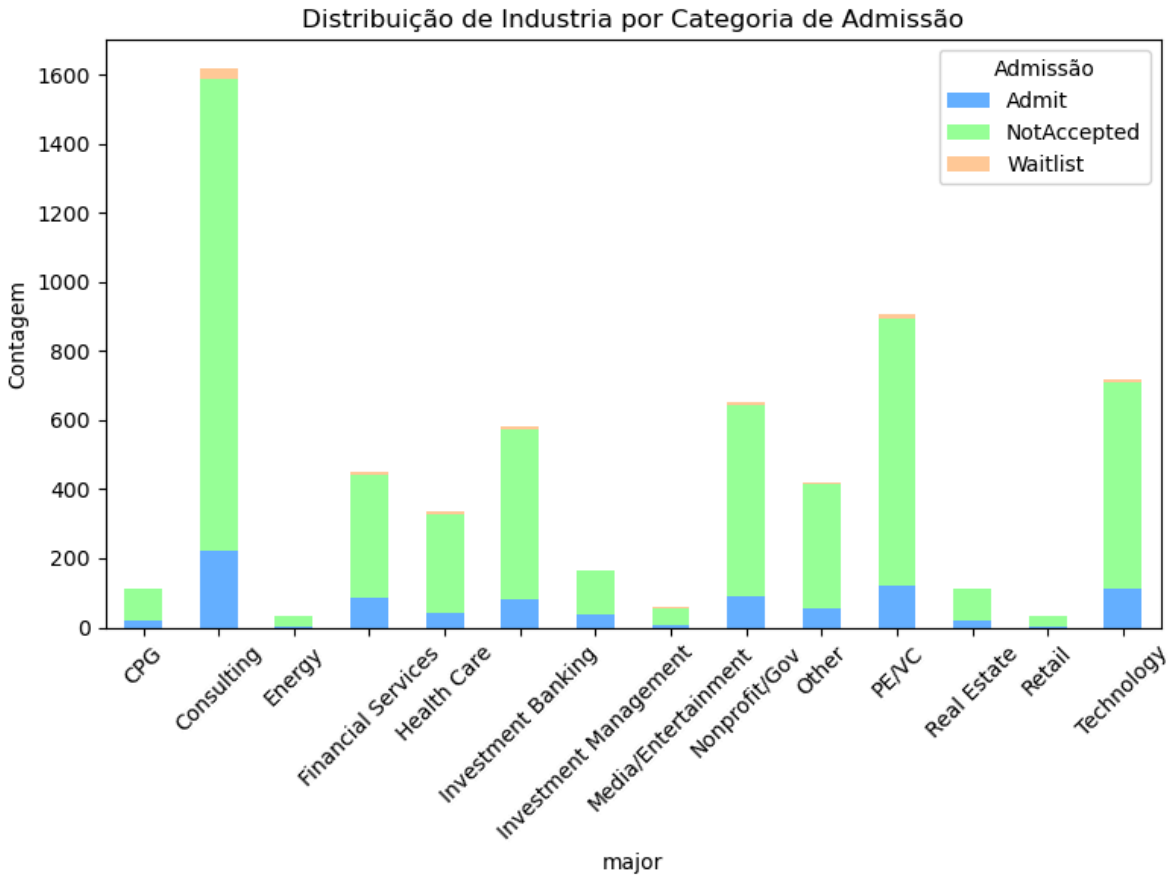
Out[113]:
```
array(['Financial Services', 'Investment Management', 'Technology',
       'Consulting', 'Nonprofit/Gov', 'PE/VC', 'Health Care',
       'Investment Banking', 'Other', 'Retail', 'Energy', 'CPG',
       'Real Estate', 'Media/Entertainment'], dtype=object)
```

In [114…
```
# Contar as frequências por Gênero e Categoria de Admissão
admissao_genero = dados.groupby(['work_industry', 'admission']).size().unstack(fill

# Criar o gráfico de barras empilhadas
admissao_genero.plot(kind='bar', stacked=True, color=['#66b3ff', '#99ff99', '#ffcc9
```

```
# Título e rótulos
plt.title('Distribuição de Industria por Categoria de Admissão')
plt.xlabel('major',)
plt.ylabel('Contagem')
plt.xticks(rotation=45)


# Exibir o gráfico
plt.legend(title='Admissão')
plt.tight_layout()
plt.show()
```



Obs:

Na nossa análise atual, observamos que a maior quantidade de candidatos se concentra nas indústrias de Consulting, Private Equity/Venture Capital (PE/VC) e Technology. Vamos interpretar o que isso pode significar e como isso impacta o processo de admissão.

A predominância de candidatos das indústrias de Consulting, PE/VC e Technology pode indicar que essas áreas estão atraindo profissionais que buscam desenvolver suas carreiras acadêmicas ou aumentar suas qualificações por meio de programas de pós-graduação. Essas indústrias geralmente valorizam a educação superior e podem ter uma cultura de incentivo ao desenvolvimento contínuo dos funcionários, o que explica o interesse em programas de admissão.

Com um maior número de candidatos provenientes dessas indústrias, a concorrência para as vagas pode ser significativamente mais acirrada. Isso implica que os candidatos devem destacar suas experiências e habilidades de forma mais efetiva para se diferenciarem. Os programas de admissão podem ter um foco mais acentuado em candidatos dessas áreas, pois podem oferecer uma maior taxa de retorno sobre o investimento em educação.

Portanto, é importante entender como isso pode influenciar as métricas de aceitação e a composição da turma.

Os candidatos que vêm dessas indústrias devem estar preparados para enfatizar suas experiências relevantes, suas habilidades transferíveis e como suas trajetórias profissionais os levaram a buscar um programa específico. É crucial que apresentem um forte case de candidatura que se alinhe com os objetivos do programa.

A concentração de candidatos nas indústrias de Consulting, PE/VC e Technology não apenas destaca as áreas de interesse dos profissionais, mas também oferece uma oportunidade de análise mais profunda sobre o impacto da experiência e do histórico profissional no processo de admissão. Essa dinâmica deve ser cuidadosamente considerada ao avaliar as estratégias de candidatura e os critérios de seleção do programa.

In [115…
```
dados['work_industry'].unique()
```

Out[115]:
```
array(['Financial Services', 'Investment Management', 'Technology',
       'Consulting', 'Nonprofit/Gov', 'PE/VC', 'Health Care',
       'Investment Banking', 'Other', 'Retail', 'Energy', 'CPG',
       'Real Estate', 'Media/Entertainment'], dtype=object)
```

In [116…
```
Technology = dados[(dados['work_industry'] == 'Technology')].shape[0]
PEVC = dados[(dados['work_industry'] == 'PE/VC')].shape[0]
Consulting = dados[(dados['work_industry'] == 'Consulting')].shape[0]
TechnologyAdmitida = dados[(dados['work_industry'] == 'Technology') & (dados['admis
PEVCAdmitido = dados[(dados['work_industry'] == 'PE/VC') & (dados['admission'] == '
ConsultingAdmitido = dados[(dados['work_industry'] == 'Consulting') & (dados['admis
pb = (TechnologyAdmitida/Technology) * 100
ph = (PEVCAdmitido/PEVC) * 100
ps = (ConsultingAdmitido/Consulting) * 100


print(f'Quantidade de candidatos aceitos de Technology: {TechnologyAdmitida} de {Te
print(f'Quantidade de candidatos aceitos de PE/VC: {PEVCAdmitido} de {PEVC} candida
print(f'Quantidade de candidatos aceitos de Consulting: {ConsultingAdmitido} de {Co
```

```
Quantidade de candidatos aceitos de Technology: 112 de 716 candidatos, propocional
de 15.64%, ou seja, 16 a cada 100 são aceitos
Quantidade de candidatos aceitos de PE/VC: 122 de 907 candidatos, propocional de 1
3.45%, ou seja, 13 a cada 100 não são aceitos
Quantidade de candidatos aceitos de Consulting: 224 de 1619 candidatos, propociona
l de 13.84%, ou seja, 14 a cada 100 não são aceitos
```

Obs:

Na nossa análise das indústrias que atraem candidatos para programas de admissão, notamos que Consulting, Private Equity/Venture Capital (PE/VC) e Technology não apenas concentram um número significativo de candidatos, mas também apresentam uma taxa considerável de aprovação. Essas três indústrias juntas representam aproximadamente 50% dos candidatos aprovados. Isso indica uma forte correlação entre a origem dos candidatos e suas chances de admissão. A cada duas pessoas aprovadas, uma vem de uma dessas indústrias, o que é um indicativo da eficácia da preparação e das competências que esses candidatos trazem.

Essa indústria também representa 52% do total de candidatos, o que sugere que a proporção de aprovados é bastante consistente com a quantidade de pessoas que se candidatam. presença quase equilibrada de candidatos e aprovados sugere que nossas

inferências sobre a competitividade e a qualidade dos candidatos dessas indústrias estão no caminho certo. Isso nos leva a acreditar que há um fator de preparação e alinhamento entre o perfil dos candidatos e as expectativas do programa.

In [117…
```
aprovados = (TechnologyAdmitida + PEVCAdmitido + ConsultingAdmitido)
candidatos = (Technology + PEVC + Consulting)
proporcional = ((TechnologyAdmitida + PEVCAdmitido + ConsultingAdmitido) / (Technol

print(f'Quantidade de candidatos aceitos do Top-3: {aprovados} de {candidatos} canc
```

```
Quantidade de candidatos aceitos do Top-3: 458 de 3242 candidatos, propocional de
14.13%, ou seja, 14 a cada 100 são aceitos
```

In [118…
```
# Lista de indústrias a serem excluídas
industrias_excluir = ['Consulting', 'PE/VC', 'Technology']

# Filtrar os dados para selecionar pessoas que não estão nas indústrias especificad
dados_filtrados = dados[~dados['work_industry'].isin(industrias_excluir)]
dados_filtrados.head()
```

Out[118]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work
0	1	Female	False	3.30	Business	Asian	620.0	3.0	
1	2	Male	False	3.28	Humanities	Black	680.0	5.0	I Ma
7	8	Male	True	3.02	Business	NaoEspecificado	630.0	6.0	
8	9	Male	False	3.24	Business	White	590.0	2.0	Non
13	14	Female	False	3.39	Business	Black	690.0	4.0	Non

In [119…
```
# Chama a função para realizar o teste Z
admitidos = dados_filtrados[dados_filtrados['admission'] == 'Admit'].shape[0]
total = dados_filtrados.shape[0]

teste_proporcao(aprovados, candidatos, admitidos, total)
```

```
Estatística Z: -0.9434
Valor-p: 0.3455
Não há diferença significativa entre as proporções.
```

Obs:

Significa que, ao comparar as taxas de aprovação entre os candidatos das indústrias de Consulting, PE/VC e Technology e as demais indústrias, não encontramos evidências estatísticas suficientes para afirmar que há uma diferença nas proporções de candidatos aprovados. Em termos práticos, isso sugere que a indústria em que os candidatos estão não parece ter um impacto significativo nas taxas de aprovação, pelo menos nas condições e no conjunto de dados que analisamos.

Resposta 6:

A partir dos testes realizados, podemos concluir que tanto a indústria quanto os anos de experiência não demonstram uma relação significativa com a nossa variável target, que é a admissão. Embora algumas indústrias tenham uma maior concentração de candidatos

aprovados, os testes estatísticos não indicaram uma diferença significativa nas taxas de aprovação. Isso sugere que, independentemente do setor em que os candidatos estão inseridos, suas chances de admissão não são afetadas de forma significativa pela indústria.

A correlação entre os anos de experiência e a admissão foi extremamente baixa (0.01), o que implica que não há uma associação clara entre a quantidade de experiência que um candidato possui e suas chances de ser admitido no programa.

7 - Existe alguma correlação entre a raça dos candidatos (para não internacionais) e as taxas de admissão?

Obs:

Primeiro vamos fazer um filtro, selecionando os candidatos não internacionais e depois verificaremos se existe diferença proporcional nas taxas de admissão de acordo com a raça.

In [120…
```
nativos = dados[dados['international'] == False]
nativos.head()
```

Out[120]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_industr
0	1	Female	False	3.30	Business	Asian	620.0	3.0	Financia Service
1	2	Male	False	3.28	Humanities	Black	680.0	5.0	Investmer Managemer
3	4	Male	False	3.47	STEM	Black	690.0	6.0	Technolog
4	5	Male	False	3.35	STEM	Hispanic	590.0	5.0	Consultin
5	6	Male	False	3.18	Business	White	610.0	6.0	Consultin

In [121…
```
dados['race'].unique()
```

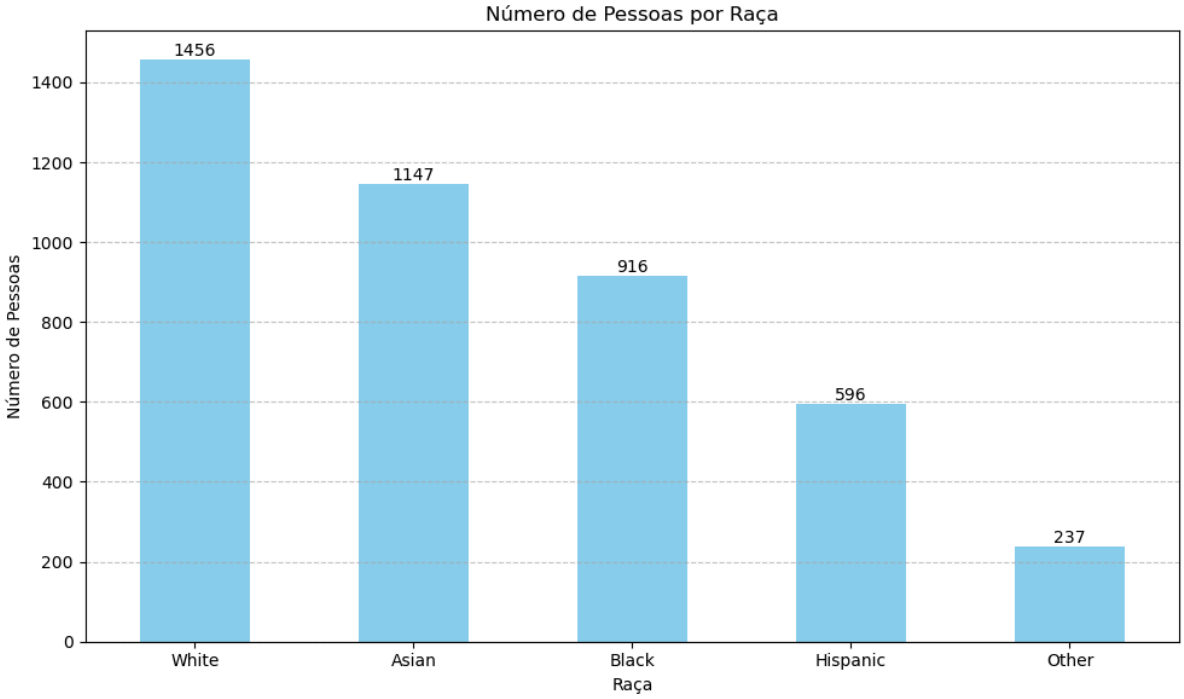
Out[121]:
```
array(['Asian', 'Black', 'NaoEspecificado', 'Hispanic', 'White', 'Other'],
      dtype=object)
```

In [122…
```
# Contar o número de pessoas com NivelEstresse 'Alto' por Indústria
contagem = nativos['race'].value_counts()

# Criar o gráfico de barras
plt.figure(figsize=(10, 6))
bars = contagem.plot(kind='bar', color='skyblue')
plt.title('Número de Pessoas por Raça')
plt.xlabel('Raça')
plt.ylabel('Número de Pessoas')
plt.xticks(rotation=0)

# Adicionar números em cima das barras
for bar in bars.patches:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
             int(bar.get_height()),
             ha='center', va='bottom')

plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()  # Ajustar layout
plt.show()
```



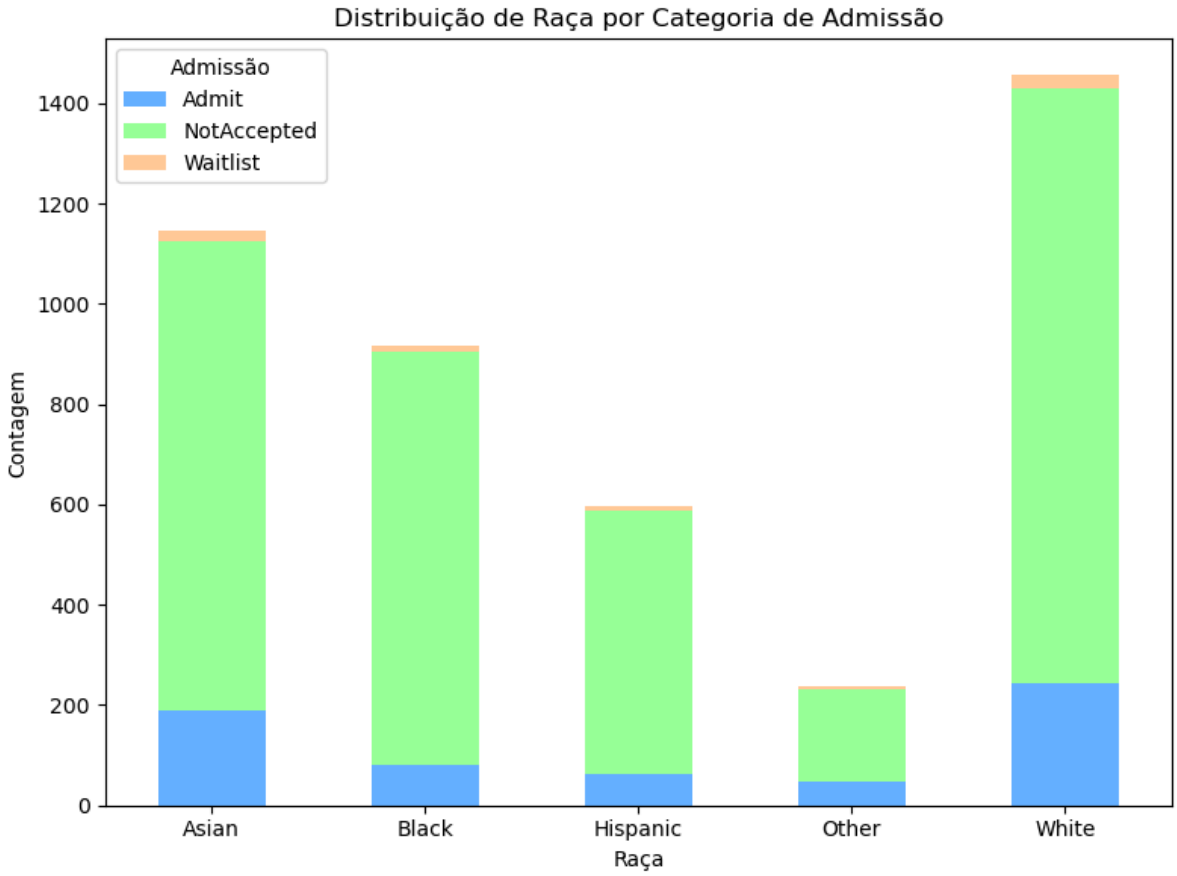
In [123…

```
# Contar as frequências por Gênero e Categoria de Admissão
admissao_genero = nativos.groupby(['race', 'admission']).size().unstack(fill_value=

# Criar o gráfico de barras empilhadas
admissao_genero.plot(kind='bar', stacked=True, color=['#66b3ff', '#99ff99', '#ffcc9

# Título e rótulos
plt.title('Distribuição de Raça por Categoria de Admissão')
plt.xlabel('Raça')
plt.ylabel('Contagem')
plt.xticks(rotation=0)


# Exibir o gráfico
plt.legend(title='Admissão')
plt.tight_layout()
plt.show()
```



Obs:

É interessante notar que os dados que estavam classificados como NaN, os quais denominamos "NaoEspecificados", estão presentes exclusivamente entre os candidatos não nativos. Essa observação pode indicar uma possível lacuna na coleta de informações sobre a raça desse grupo, o que merece atenção em nossa análise.

In [124…

```
White = dados[(dados['race'] == 'White')].shape[0]
Asian = dados[(dados['race'] == 'Asian')].shape[0]
Black = dados[(dados['race'] == 'Black')].shape[0]
Hispanic = dados[(dados['race'] == 'Hispanic')].shape[0]
Other = dados[(dados['race'] == 'Other')].shape[0]
WhiteAdmitida = dados[(dados['race'] == 'White') & (dados['admission'] == 'Admit')]
AsianAdmitido = dados[(dados['race'] == 'Asian') & (dados['admission'] == 'Admit')]
BlackAdmitido = dados[(dados['race'] == 'Black') & (dados['admission'] == 'Admit')]
HispanicAdmitido = dados[(dados['race'] == 'Hispanic') & (dados['admission'] == 'Ad
OtherAdmitido = dados[(dados['race'] == 'Other') & (dados['admission'] == 'Admit')]
pw = (WhiteAdmitida/White) * 100
pa = (AsianAdmitido/Asian) * 100
pb = (BlackAdmitido/Black) * 100
ph = (HispanicAdmitido/Hispanic) * 100
po = (OtherAdmitido/Other) * 100


print(f'Quantidade de candidatos aceitos de White: {WhiteAdmitida} de {White} candi
print(f'Quantidade de candidatos aceitos de Asian: {AsianAdmitido} de {Asian} candi
print(f'Quantidade de candidatos aceitos de Black: {BlackAdmitido} de {Black} candi
print(f'Quantidade de candidatos aceitos de Hispanic: {HispanicAdmitido} de {Hispar
print(f'Quantidade de candidatos aceitos de Other: {OtherAdmitido} de {Other} candi
```

```
Quantidade de candidatos aceitos de White: 244 de 1456 candidatos, propocional de
16.76%, ou seja, 17 a cada 100 são aceitos
Quantidade de candidatos aceitos de Asian: 190 de 1147 candidatos, propocional de
16.56%, ou seja, 17 a cada 100 não são aceitos
Quantidade de candidatos aceitos de Black: 80 de 916 candidatos, propocional de 8.
73%, ou seja, 9 a cada 100 não são aceitos
Quantidade de candidatos aceitos de Hispanic: 62 de 596 candidatos, propocional de
10.40%, ou seja, 10 a cada 100 não são aceitos
Quantidade de candidatos aceitos de Other: 46 de 237 candidatos, propocional de 1
9.41%, ou seja, 19 a cada 100 não são aceitos
```

Obs:

Observamos que a taxa de aceitação varia significativamente entre os diferentes grupos raciais. Enquanto os candidatos brancos e asiáticos têm taxas de aceitação semelhantes, os candidatos negros e hispânicos apresentam taxas consideravelmente mais baixas. A categoria "Outros" mostra a maior proporção de aceitação, embora a amostra seja menor. Essa discrepância nas taxas de aceitação pode indicar a necessidade de uma análise mais aprofundada sobre as políticas de admissão e seu impacto em diferentes grupos raciais.

Resposta 7:

A análise dos dados revela uma diferença significativa nas taxas de admissão quando observamos os grupos raciais. Essa disparidade pode refletir as desigualdades estruturais que historicamente impactam a população negra, que frequentemente enfrenta desafios socioeconômicos maiores e menor acesso a oportunidades em comparação com outros grupos.

Enquanto candidatos brancos e asiáticos apresentam taxas de aceitação mais altas, a população negra mostra uma proporção de aceitação consideravelmente mais baixa, evidenciando como esses grupos estão frequentemente em posições marginalizadas na sociedade. Essa situação não é apenas uma questão de performance individual, mas também de acesso desigual a recursos, informações e oportunidades.

Essa desigualdade pode ser reflexo de uma série de fatores, incluindo a falta de conhecimento sobre as oportunidades disponíveis, menor acesso a redes de apoio ou mesmo barreiras financeiras que dificultam a competitividade. Em muitos casos, as oportunidades sequer chegam a essas comunidades, reforçando um ciclo de exclusão. É fundamental, portanto, que sejam desenvolvidas estratégias e políticas que visem reduzir essas disparidades, garantindo que todos os grupos tenham acesso igualitário às oportunidades que lhes são de direito.

Modelagem

Obs:

O modelo de Árvore de Decisão é um dos algoritmos de aprendizado supervisionado mais simples e interpretáveis, amplamente usado para tarefas de classificação e regressão.

Uma árvore de decisão é uma estrutura de modelo que divide repetidamente o conjunto de dados em subconjuntos baseados em um critério de decisão, criando uma estrutura semelhante a uma árvore, com nós de decisão e folhas.

Nó Raiz (Root Node): O ponto de partida da árvore, que contém todos os dados e representa a primeira decisão a ser tomada. Nós Internos (Decision Nodes): Pontos de decisão que dividem os dados com base em uma característica ou condição. Folhas (Leaf Nodes): Resultados finais das ramificações, que representam a classe ou valor predito.

Como o Modelo Funciona?

O modelo escolhe as melhores divisões dos dados usando critérios de pureza, como:

Gini Impurity: Mede a impureza de um nó com base na probabilidade de uma amostra ser classificada incorretamente. Entropia (Information Gain): Mede o ganho de informação ao dividir um nó, preferindo divisões que aumentam a "ordem" dos dados. Reduction in Variance: Usado em problemas de regressão para minimizar a variação nos nós filhos em comparação ao nó pai.

Treinamento do Modelo:

Durante o treinamento, o modelo analisa as features (características) dos dados e cria regras de decisão para dividir os dados da forma que melhor separe as classes alvo. Cada divisão é escolhida para maximizar a separação entre as classes

Esse modelo é ideal quando se busca uma abordagem explicativa e fácil de entender, embora, para dados complexos, possa ser interessante usar versões avançadas como Random Forest ou métodos de boosting que superam algumas das limitações das árvores de decisão simples

In [125…
```
# Supondo que X seja o dataframe com as features e y seja o target
# Dividindo os dados em treino e teste

X = dados.drop(columns={'application_id', 'admission'})
y= dados['admission']
```

In [126…
```
colunas_categoricas = ['gender', 'international', 'major','race','work_industry']

# Inicializa o encoder
label_encoder = LabelEncoder()

# Aplica o Label Encoding para cada coluna categórica
for coluna in colunas_categoricas:
    X[coluna] = label_encoder.fit_transform(X[coluna])
```

In [127…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

Obs:

Label Encoding é uma técnica de pré-processamento de dados usada para converter variáveis categóricas em valores numéricos, tornando-as mais adequadas para modelos de machine learning. Embora o One-Hot Encoding seja geralmente preferível para variáveis sem ordem intrínseca, pois cria colunas binárias para cada categoria e evita que o modelo interprete uma ordem numérica inexistente, optaremos pelo uso do Label Encoding neste estudo para simplificar o entendimento do processo.

In [128…
```
# Criando o modelo de árvore de decisão
modelo_arvore = DecisionTreeClassifier(random_state=42)
```

```
# Treinando o modelo
modelo_arvore.fit(X_train, y_train)
```

Out[128]:

DecisionTreeClassifier

DecisionTreeClassifier(random_state=42)

In [129…

```
# Fazendo previsões com o conjunto de teste
previsoes = modelo_arvore.predict(X_test)

# Avaliando o modelo
acuracia = accuracy_score(y_test, previsoes)
print(f"Acurácia do modelo: {acuracia:.2f}")
```

```
Acurácia do modelo: 0.79
```

In [130…

```
# Relatório de classificação
print("\nRelatório de Classificação:")
print(classification_report(y_test, previsoes))
```

```
Relatório de Classificação:
              precision    recall  f1-score   support

       Admit       0.40      0.38      0.39       196
 NotAccepted       0.87      0.88      0.88      1025
    Waitlist       0.04      0.06      0.05        18

    accuracy                           0.79      1239
   macro avg       0.44      0.44      0.44      1239
weighted avg       0.79      0.79      0.79      1239
```

In [131…

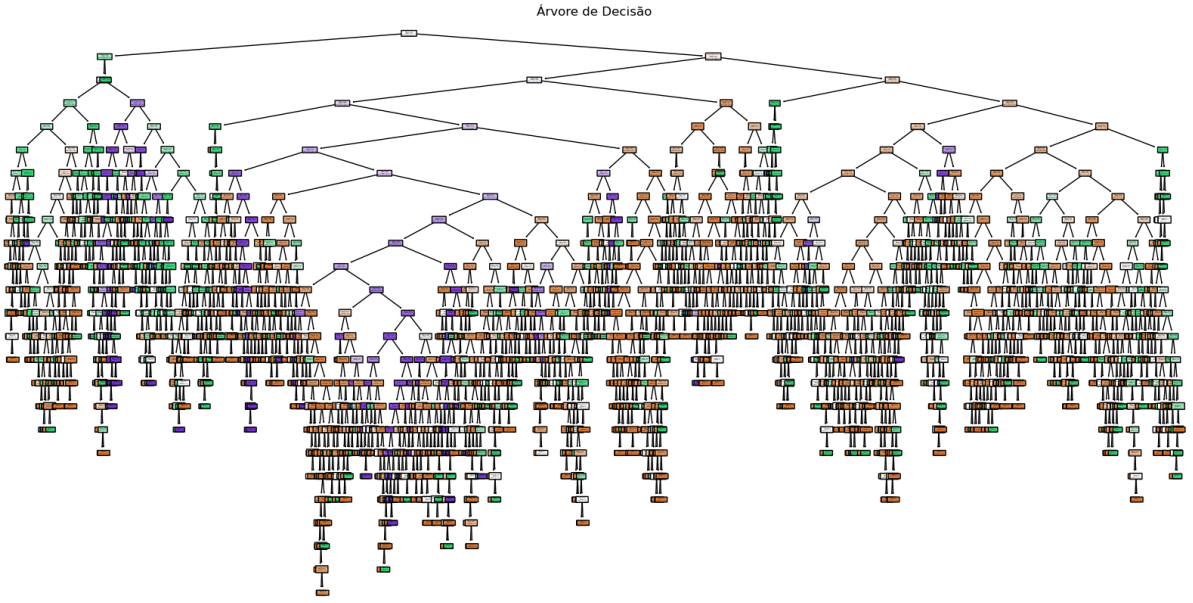
```
f1_ponderado = f1_score(y_test, previsoes, average='weighted')
f1_ponderado
```

Out[131]:

```
0.7861530148100372
```

In [139…

```
# Visualização da árvore de decisão
plt.figure(figsize=(20, 10))
tree.plot_tree(modelo_arvore, filled=True, feature_names=X.columns, class_names=['N
plt.title('Árvore de Decisão')
plt.show()
```



Obs:

O modelo apresenta um desequilíbrio notável, mostrando um desempenho satisfatório na classe majoritária ("NotAccepted"), enquanto enfrenta dificuldades significativas em prever corretamente as classes minoritárias ("Admit" e "Waitlist"). Essa disparidade indica a necessidade de ajustes, como a aplicação de técnicas de balanceamento de classes ou métodos de modelagem mais robustos, que possam lidar eficazmente com a desproporção dos dados. Embora a acurácia geral do modelo seja razoável, alcançando 0.78, é essencial considerar que essa métrica não reflete adequadamente o desempenho nas classes de interesse.

Ao focarmos na previsão de quem será admitido, poderíamos optar por abordagens que priorizem a precisão nessa classe específica, mesmo que isso resulte em uma leve redução na acurácia geral. Essa estratégia não só aumentaria a capacidade do modelo de identificar candidatos admitidos, mas também proporcionaria insights valiosos para a tomada de decisões nas admissões.

Em suma, o estudo revelou áreas críticas para aprimoramento e destaca a importância de uma modelagem mais equilibrada e direcionada, garantindo que as decisões baseadas nos resultados sejam mais informadas e efetivas. Com os ajustes adequados, o modelo pode se tornar uma ferramenta poderosa para prever admissões, contribuindo significativamente para a estratégia de seleção.

Vamos ustilizar a SMOTEN, que é uma técnica de oversampling da biblioteca imbalanced-learn (imblearn) que é uma variação do método SMOTE (Synthetic Minority Over-sampling Technique). A principal diferença do SMOTEN em relação ao SMOTE tradicional é que ele é especificamente projetado para lidar com variáveis categóricas.

A finalidade consiste em gerar novos exemplos sintéticos para classes minoritárias, focando em variáveis categóricas (nominais) que não possuem uma ordem intrínseca, criando amostras sintéticas calculando o modo das categorias dos vizinhos mais próximos, sem gerar valores intermediários como o SMOTE faz com variáveis numéricas.

O SMOTEN é particularmente útil quando se trabalha com dados categóricos que precisam ser balanceados, oferecendo uma solução mais adequada que o SMOTE tradicional para esses casos. Ele ajuda a melhorar o desempenho dos modelos, especialmente em tarefas onde o desbalanceamento das classes pode prejudicar significativamente a qualidade das previsões.

In [133…
```
print(f"Original class counts: {Counter(y)}")
sampler = SMOTEN(random_state=22)
X_res, y_res = sampler.fit_resample(X, y)
print(f"Class counts after resampling {Counter(y_res)}")
```

```
Original class counts: Counter({'NotAccepted': 5194, 'Admit': 900, 'Waitlist': 10
0})
Class counts after resampling Counter({'Admit': 5194, 'NotAccepted': 5194, 'Waitli
st': 5194})
```

In [134…
```
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.2, ra
```

In [135…
```
# Criando o modelo de árvore de decisão
modelo_arvore = DecisionTreeClassifier(random_state=42)

# Treinando o modelo
modelo_arvore.fit(X_train, y_train)
```

Out[135]:
```
DecisionTreeClassifier(random_state=42)
```

In [136…
```
# Fazendo previsões com o conjunto de teste
previsoes = modelo_arvore.predict(X_test)

# Avaliando o modelo
acuracia = accuracy_score(y_test, previsoes)
print(f"Acurácia do modelo: {acuracia:.2f}")
```

```
Acurácia do modelo: 0.91
```

In [137…
```
# Relatório de classificação
print("\nRelatório de Classificação:")
print(classification_report(y_test, previsoes))
```

```
Relatório de Classificação:
              precision    recall  f1-score   support

       Admit       0.87      0.90      0.89      1095
 NotAccepted       0.89      0.84      0.86      1019
    Waitlist       0.96      0.98      0.97      1003

    accuracy                           0.91      3117
   macro avg       0.91      0.91      0.91      3117
weighted avg       0.91      0.91      0.91      3117

              precision    recall  f1-score   support

       Admit       0.87      0.90      0.89      1095
 NotAccepted       0.89      0.84      0.86      1019
    Waitlist       0.96      0.98      0.97      1003

    accuracy                           0.91      3117
   macro avg       0.91      0.91      0.91      3117
weighted avg       0.91      0.91      0.91      3117
```

In [138…
```
f2_ponderado = f1_score(y_test, previsoes, average='weighted')
f2_ponderado
```

Out[138]:
```
0.9067277491993692
```

Conclusão

Após a aplicação do SMOTE, nosso modelo apresentou uma melhora significativa em seu desempenho. Inicialmente, tínhamos uma acurácia de 0.79 e um F1-score ponderado de 0.78, valores que indicavam uma boa performance, mas ainda insuficientes para capturar adequadamente as classes minoritárias. Com o uso do oversampling via SMOTE, conseguimos elevar o F1-score para 0.90 e a acurácia para 0.91. Essa abordagem se mostrou eficaz para lidar com o desbalanceamento dos dados, melhorando a capacidade do modelo de identificar corretamente as classes durante os testes. Esses resultados reforçam que o uso

de técnicas de balanceamento pode ser crucial para aumentar a precisão e a confiabilidade dos modelos preditivos, especialmente em contextos com classes desiguais.