Análise de Dados – Exploratória e ML

Sumário

[Introdução 4](#_Toc181787977)

[Objetivo do aplicativo 4](#_Toc181787978)

[Objetivo da Análise Exploratória de Dados 4](#_Toc181787979)

[Objetivo da Análise de Dados (para Aprendizado de Máquina) 5](#_Toc181787980)

[Levantamento dos dados na análise exploratória 5](#_Toc181787981)

[Busca dos dados 5](#_Toc181787982)

[Justificativa de uso 6](#_Toc181787983)

[Descrição da base de dados de trabalho 6](#_Toc181787984)

[Limpeza dos dados 10](#_Toc181787985)

[Condicionamento para alimentar o modelo de ML 14](#_Toc181787986)

[Condicionamento inicial 14](#_Toc181787987)

[Definição dos objetivos e das classes 15](#_Toc181787988)

[Objetivos Específicos: 15](#_Toc181787989)

[Listar e descrever a(s) resposta(s) (y) e as classes que cada uma dessas respostas vai prever. 16](#_Toc181787990)

[Definição dos Modelos Mais Adequados para Analisar os Dados 16](#_Toc181787991)

[Visão Geral dos Modelos Considerados 17](#_Toc181787992)

[Critérios de Seleção dos Modelos 17](#_Toc181787993)

[Descrição dos modelos selecionados 18](#_Toc181787994)

[Modelo 1: K-Nearest Neighbors (KNN) 18](#_Toc181787995)

[Modelo 2: Árvore de Decisão (Gini e Entropy) 18](#_Toc181787996)

[Modelo 3: Naive Bayes 19](#_Toc181787997)

[Aplicação dos Modelos Selecionados 20](#_Toc181787998)

[Divisão dos Dados 20](#_Toc181787999)

[Treinamento dos Modelos 20](#_Toc181788000)

[Métricas de Avaliação 21](#_Toc181788001)

[Análise dos Resultados 21](#_Toc181788002)

[Modelo 1: Árvore de Decisão com Critério Gini 21](#_Toc181788003)

[Modelo 2: Árvore de Decisão com Critério Entropia 22](#_Toc181788004)

[Modelo 3: Naive Bayes 22](#_Toc181788005)

[Modelo 4: K-Nearest Neighbors (KNN) 22](#_Toc181788006)

[Comparação Direta 23](#_Toc181788007)

[Forças e Fraquezas 23](#_Toc181788008)

[Ajustes Necessários 24](#_Toc181788009)

[Impacto das Modificações nos Modelos 26](#_Toc181788010)

[Modelo 1: GaussianNB 26](#_Toc181788011)

[Modelo 2: KNN 26](#_Toc181788012)

[Modelo 3: DecisionTreeClassifier 26](#_Toc181788013)

[Comparação Entre Modelos 27](#_Toc181788014)

[Conclusão 28](#_Toc181788015)

[Conclusão Final 29](#_Toc181788016)

[RPA: descrição do funcionamento e código python 30](#_Toc181788017)

[**Script de Modelagem de Dados - Script\_Norm.py** 32](#_Toc181788018)

[Modelo no APP: descrição do funcionamento do modelo e explicação de como foi publicado no app 33](#_Toc181788019)

[**Aplicação do modelo:** 33](#_Toc181788020)

Análise de Dados para *O App BiMo*

Startup: Kyndred

Integrantes:

Davi de Siqueira Cavalcante | n° 3 | 2°H Tech

Marcus Vinicius Rigueto Thomazetti | n°13 | 2°H Tech

Link do repositório com a criação do modelo e o Data Cleaning:

<https://github.com/BiMODados/AED_AD.git>

# Introdução

Nosso aplicativo foi desenvolvido para atender às necessidades de microempreendedores (MEIs) e empresas de pequeno porte que enfrentam desafios em seus negócios, como a venda de maquinário, a escassez de mão de obra, a busca por novos clientes e a aquisição de novos conhecimentos e experiências. Em muitos casos, esses empreendedores têm dificuldade em acessar uma plataforma que possa atender às suas necessidades de forma rápida e eficiente, o que pode resultar em oportunidades perdidas ou na subutilização de recursos, como equipamentos e serviços especializados. Essa lacuna no mercado atual impacta diretamente o crescimento e a sustentabilidade desses negócios, especialmente em momentos de mudança ou encerramento das atividades.

O público-alvo do aplicativo são MEIs, microempresas e pequenas empresas que buscam uma plataforma dedicada à venda de maquinário e à prestação de serviços. Nosso objetivo é facilitar o acesso a oportunidades, permitindo que esses empreendedores se conectem de maneira mais estratégica, compartilhem recursos e ampliem sua rede de contatos, tudo dentro de um ambiente voltado para atender às especificidades de negócios de pequeno porte.

## Objetivo do aplicativo

O objetivo principal do nosso aplicativo é fornecer uma rede de apoio integrada que resolva os desafios enfrentados por microempreendedores individuais (MEIs) e pequenas empresas. Através de uma plataforma de comunicação dedicada, o aplicativo conecta esses empreendedores a oportunidades de venda de maquinário, contratação de mão de obra e prestação de serviços especializados por meio de um fórum aberto. Além disso, o aplicativo funciona como um Marketplace, onde os usuários podem listar e adquirir equipamentos necessários para seus negócios, permitindo uma troca eficiente de recursos. Para promover o desenvolvimento contínuo, a plataforma também oferece cursos e treinamentos organizados por nossa equipe, ajudando os empreendedores a adquirirem novos conhecimentos e habilidades. Dessa forma, o aplicativo facilita o crescimento sustentável dessas empresas, oferecendo uma solução completa e de fácil acesso para suas necessidades operacionais e de capacitação.

## Objetivo da Análise Exploratória de Dados

Montar, analisar e limpar um conjunto de dados relacionado ao aplicativo para ser utilizado em análise de dados exploratória e para fins de aprendizado de máquina (ML):

O objetivo da Análise Exploratória para nosso projeto, é preparar e refinar um conjunto de dados, para gerar insights estratégicos e suporte ao desenvolvimento dos modelos de aprendizado de máquina (ML) para o nosso app. Além de entender melhor o contexto das MEIs e MEs do Brasil por meio de dados públicos

Essa base de dados contém informações detalhadas sobre empresas brasileiras, incluindo microempreendedores (MEIs) e pequenas empresas, que são nosso público-alvo. A partir dessa base, realizamos uma análise que nos permita entender melhor a base e suas características que possam ajudar na confecção dos serviços prestados pelo aplicativo. Esse processo de análise e limpeza de dados é fundamental para garantir que as funcionalidades oferecidas atendam de forma eficaz às necessidades dos usuários, oferecendo um ambiente digital bem adaptado às especificidades das MEIs e MEs. Pois o intuito do app é ser um grande suporte para as MEIs e Mês de todo brasil, fornecendo uma rede de apoio e network para eles.

## Objetivo da Análise de Dados (para Aprendizado de Máquina)

O objetivo da Análise de Dados com modelos de Aprendizado de Máquina é prever a situação cadastral das MEIs e MEs, classificando-as como Ativas, Encerradas (Baixadas ou Nulas) ou Irregulares (Inaptas ou Suspensas). Para alcançar essa previsão, criamos três modelos diferentes: KNN, Decision Tree e Naive Bayes.

Para otimizar a escolha do modelo final, desenvolvemos uma pipeline que possibilitará comparar o desempenho desses três modelos. Nesse processo, aplicaremos quatro tipos de normalização aos dados e testaremos o impacto de cada uma no desempenho dos modelos. Ao final dos testes, seguiremos em frente com o modelo que apresentar o melhor desempenho, garantindo uma previsão mais eficiente da situação cadastral das MEIs, o que permitirá ao aplicativo prever se uma MEI ou ME acabará fechado ou não e assim agir da melhor forma possível para evitar isso, por meio de recomendações de serviços ou cursos para que nossos clientes possam adquirir novos conhecimentos para guiar sua MEI pelo melhor caminho possível, de acordo com seu contexto atual.

# Levantamento dos dados na análise exploratória

## Busca dos dados

Para o levantamento de dados, fizemos uma pesquisa minuciosa em diversos sites e plataformas para encontrar a base identificamos e selecionamos a base de dados pública do Cadastro Nacional de Pessoas Jurídicas (CNPJ), disponível no portal [dados.gov.br](https://dados.gov.br), Link da base:

<https://dados.gov.br/dados/conjuntos-dados/cadastro-nacional-da-pessoa-juridica---cnpj>

Essa base, que atende plenamente às nossas necessidades, ela foi escolhida devido ao seu foco em informações cadastrais de empresas brasileiras, incluindo microempreendedores individuais (MEIs) e microempresas (MEs), que constituem o público-alvo do nosso aplicativo.

A base do CNPJ oferece dados completos, um histórico, e um volume muito grande de dados sobre a situação cadastral, natureza jurídica, porte das empresas, além de informações geográficas e econômicas que dão muito valor para nossa análise. Esse conjunto de dados nos permite explorar e identificar padrões e variáveis relevantes para o desenvolvimento de modelos de aprendizado de máquina, com o objetivo de prever a situação cadastral das MEIs e MEs. Dessa forma, essa base governamental se mostra como uma fonte confiável por ser diretamente do dados.gov.br e abrangente, possibilitando para nós uma fonte de dados completas, ideal para nossas análises.

Em relação a questionários, realizamos um levantamento focado no aplicativo e no contexto de MEIs e MEs, coletando informações sobre desafios e necessidades desse público. Contudo, os dados obtidos nesse questionário foram utilizados para disciplina Business Intelligence, para aprofundar o contexto de atuação do aplicativo, nós decidimos não aplicar diretamente na matéria de Análise Exploratória de Dados. Dessa forma, para a AED, utilizamos exclusivamente os dados da base do CNPJ, que oferece um panorama confiável e abrangente do setor.

## Justificativa de uso

A escolha da base pública do CNPJ, disponível no dados.gov.br, é essencial para o trabalho, pois oferece dados completos e confiáveis sobre o público-alvo do aplicativo: microempreendedores individuais (MEIs) e microempresas (MEs) no Brasil. Esses dados, que incluem situação cadastral, natureza jurídica, porte, informações geográficas e econômicas, além de um histórico detalhado, permitem uma análise precisa das condições cadastrais dessas empresas. Essa base é ideal para desenvolver modelos de aprendizado de máquina que visam prever a situação cadastral de MEIs e MEs, contribuindo para a segmentação e o atendimento mais estratégico do público-alvo. A confiabilidade dos dados governamentais, obtidos de uma fonte oficial, garante que o trabalho seja embasado em informações de qualidade, possibilitando insights aplicáveis e alinhados ao foco do aplicativo em solucionar desafios dos pequenos empreendedores.

## Descrição da base de dados de trabalho

Análises sobre a base

* Tipos dos dados por coluna:

CNPJ Básico int64

CNPJ Ordem int64

CNPJ DV int64

MATRIZ/FILIAL int64

Nome Fantasia object

Situação Cadastral int64

Data Situação Cadastral int64

Motivo Situação Cadastral int64

Nome da Cidade no Exterior float64

Pais float64

Data Início de Atividade int64

CNAE Fiscal Principal int64

CNAE Fiscal Secundária object

Tipo Logradouro object

Logradouro object

Número object

Complemento object

Bairro object

CEP int64

Município int64

DDD 1 float64

Telefone 1 object

DDD 2 float64

Telefone 2 float64

DDD Do Fax float64

Fax object

Correio Eletrônico object

Razão Social object

Natureza Jurídica int64

Qualificação do Responsável int64

Porte Empresa float64

Capital Social float64

Ente Federativo object

id\_mun int64

municipios object

id\_cnaes int64

cnaes object

id\_mot int64

motivos object

id\_nat int64

Natureza Jurídica object

**- Valores-limite: máximo e mínimo para dados numéricos, lista de valores para dados categóricos:**

Valores-limite:

Campo Mínimo Máximo

Capital Social 0.00 52,101,911,486.00

Lista de valores para dados categóricos:



Fica inviável colocar aqui a lista de valores para dados categóricos, pois nossa base antes da limpeza tem 42 colunas, algumas delas com mais de 400 valores diferentes, essas informações se encontram no notebook jupyter anexado (Data Cleaning AED)

* Plotagem dos gráficos de análise preliminar dos dados brutos usando Python:

- A plotagem dos gráficos brutos em Python está no arquivo jupyter anexado ou no link:

<Link do repositório>

* Parâmetros estatísticos que descrevem os dados: média, moda, desvio padrão, variância etc.

Resumo estatístico da coluna Capital Social:

Média (R$5.943.128,52): Embora a média seja de aproximadamente R$5,9 milhões, ela está altamente influenciada por outliers extremamente altos. Esse efeito indica que a maioria dos valores é, na verdade, muito menor do que a média sugere.

Mediana (R$5,0): A mediana, o ponto central dos dados, é R$5,0. Isso significa que metade das empresas tem um capital social menor ou igual a R$5,0. Em contraste com a média, a mediana revela que a maioria dos valores é bastante baixa, o que reforça que os dados estão concentrados em valores pequenos, enquanto poucos valores muito altos aumentam a média.

Moda (R$5,0): O valor mais comum para o capital social é R$5,0, mostrando que várias empresas estão concentradas nesse valor exato, possivelmente devido a práticas comuns de registro de capital social mínimo.

Desvio Padrão (R$463.017.042,72): O desvio padrão é muito alto, o que indica uma variação significativa em relação à média. Esse valor reflete que os dados estão muito espalhados, e que existem valores extremamente altos que estão longe da média, contribuindo para a grande dispersão.

Variância (2,1438e+17): A variância confirma essa dispersão ao quadrado em torno da média, reforçando que a distribuição dos valores é altamente dispersa e que muitos dados estão afastados da média.

Mínimo e Máximo (R$0,0 e R$52.101.911.486,0): O capital social mínimo é zero, enquanto o valor máximo é de R$52,1 bilhões. Essa grande diferença entre o menor e o maior valor mostra que há uma enorme variação no capital social entre as empresas, desde empresas sem capital até empresas com valores de capital social absurdos.

Amplitude (R$52.101.911.486,0): A amplitude, que é a diferença entre o máximo e o mínimo, é extremamente grande, indicando uma variação total gigantesca nos valores de capital social.

Curtose (11.938,28): A alta curtose significa que a distribuição dos valores de capital social é muito achatada e apresenta muitos valores extremos, ou outliers, que se desviam bastante do centro. Esses outliers estão provavelmente contribuindo para os valores altos da média e do desvio padrão.

Assimetria (106,82): A assimetria positiva muito alta indica que a distribuição é fortemente inclinada para a direita. Em outras palavras, a maioria dos valores está concentrada em valores baixos, mas há uma cauda longa de valores extremamente altos que distorce a distribuição.

Sempre indique como fez a análise, seja com seu código Python próprio, ou dizendo qual a biblioteca de análise exploratória de dados (por exemplo, ydata-profiling).

Análise dos tipos das colunas:

Código:

Forma

Descrição gerada automaticamente com confiança média

Libs:

pandas

Valores mínimos e máximos:

Código:

Texto

Descrição gerada automaticamente

Libs:

Pandas

Análise valores categóricos:

Código:

Texto

Descrição gerada automaticamente

Libs:

Pandas

Código:

Texto

Descrição gerada automaticamente

Libs:

pandas

scipy.stats

kurtosis

skew

No relatório, coloque os resultados da análise dos dados brutos (parâmetros estatísticos e gráficos). O código (notebook) deve ser fornecido em um anexo ou como um link para o GitHub:

Código completo:

Disponível no arquivo DataCleaning.ipynb <- acesso este arquivo pelo link do github acima.

Para plotagem dos dados brutos encontrados, monte um dashboard, de acordo com os requisitos da disciplina Business Intelligence (aproveite o que for desenvolvido para essa disciplina).

O dashboard para a plotagem dos dados brutos foi desenvolvido diretamente em Python, considerando o tamanho da base, que tornou inviável sua transferência para a ferramentas Power BI. Optamos por trabalhar em Python para garantir uma análise eficiente e detalhada dos dados, permitindo visualizar todos os parâmetros relevantes da base na nossa visão.

O dashboard dos dados brutos se encontra disponível no arquivo anexado:

Graficos\_base\_gov.ipynb.

# Limpeza dos dados

- Descrição da Limpeza da Base:

Como descrito no levantamento dos dados nós baixamos a base no dados.gob.br e lá ela estava dividida em diversos arquivos do tipo txt, nós baixamos esses arquivos e transformamos eles em arquivos excel, como mostrado no print a seguir:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Assim, como a base está dividida em dezenas de arquivos, tivemos o trabalho de juntar as bases em um único DataFrame para podermos começar com a limpeza dos dados:

Descrição do procedimento de limpeza e preparação dos dados para poder prosseguir para a análise preditiva:

**Remoção de colunas e motivos:**

Como nossa base inicialmente havia 42 colunas, nós tivemos que remover diversas colunas, como mostraremos a seguir:

**- Identificadores e Dados Técnicos:**

'CNPJ Básico', 'CNPJ Ordem', 'CNPJ DV', 'MATRIZ/FILIAL', 'Ente Federativo', 'id\_mun', 'id\_cnaes', 'id\_mot', 'id\_nat', 'municipios':

motivo: Foram removidas pois representam códigos e identificadores específicos, são úteis para controle interno e organização, mas sem valor preditivo direto para o modelo.

**- Informações Geográficas muito especificas e de Contato:**

'Nome da Cidade no Exterior', 'Pais', 'Tipo Logradouro', 'Logradouro', 'Número', 'Complemento', 'Bairro', 'CEP', 'DDD 1', 'Telefone 1', 'DDD 2', 'Telefone 2', 'DDD Do Fax', 'Fax', 'Correio Eletrônico'

Motivo: Foram excluídos pois tratam de informações de localização especificas demais e contatos que, embora detalhadas, não são relevantes para a previsão da situação cadastral.

**- Outras Informações Cadastrais:**

'Nome Fantasia', 'Razão Social', 'Qualificação do Responsável', 'Motivo Situação Cadastral', 'Data Situação Cadastral'

Motivos: Foram removidas por fornecerem detalhes que não influenciam diretamente a previsão. Especificamente a Data Situação Cadastral estava com dados completamente iguais, todos os dados dessa coluna eram '01/01/1970'

**- Dados Redundantes ou Altamente Variáveis:**

'CNAE Fiscal Principal', 'CNAE Fiscal Secundária'

Motivos: Foram descartadas por serem dados categóricos de alta granularidade. A previsão do modelo pode se beneficiar mais de um nível consolidado de informações (como mantido em 'cnaes'), eliminando a complexidade adicional que esses campos podem causar.

**- Colunas que mantivemos:**

Mantivemos apenas as colunas essenciais como 'Situação Cadastral', 'Data Inicio de Atividade', 'UF', 'Porte Empresa', 'Capital Social', 'municipios', 'cnaes', e 'Natureza Juridica', que fornecem informações estratégicas, consolidadas e diretamente relevantes para a análise e previsão do modelo de IA.

**- Mudança de tipos e formatações das colunas que seguiram a diante:**

Data de Início de Atividade:

Originalmente, a coluna estava no formato de texto (string). Então nós a convertemos para o tipo datetime para facilitar as manipulações com ela. Em seguida, a dividimos em três colunas distintas para representar o ano, mês e dia, permitindo uma análise temporal mais detalhada.

**Situação Cadastral:**

Esta coluna estava originalmente codificada (ex.: 02 - NULA, 08 - ATIVA, 01 - SUSPENSA, 04 - INAPTA, 03 - BAIXADA). Aplicamos um mapeamento para transformar os códigos em strings legíveis, como “Ativa”, “Inapta”, “Suspensa” etc., melhorando a interpretação e a leitura dessa coluna.

**Junção de Classes de Situação Cadastral:**

Com base na análise gráfica, vimos que algumas classes tinham pouquíssimas ocorrências em comparação com outras, tendo isso em mente, observamos que algumas categorias tinham similaridades funcionais e de status, então decidimos fazer à fusão das seguintes classes:

**- Ativa:** permaneceu como “Ativa”.

**- Inapta e Suspensa:** foram combinadas em “Irregular” devido à similaridade do funcionamento das MEIs nesse estado, no caso o bloqueio de algumas ações, como emitir notas fiscais.

**- Baixada e Nula:** fundidas em “Encerrada”, pois em ambas as situações a MEI encerra seu funcionamento, sendo baixada por vontade própria do dono ou a pedido da receita federal e nula por algum tipo de corrupção ou ações que vão contra a lei.

Gráfico antes da junção:

Gráfico, Gráfico de pizza

Descrição gerada automaticamente

Gráfico após a junção:

Gráfico, Gráfico de pizza

Descrição gerada automaticamente

Porte da Empresa: Similar ao processo da coluna "Situação Cadastral", os valores estavam codificados (0 - Não Informado, 1 - Microempresa, 3 - Empresa de Pequeno Porte). Realizamos a transformação para strings para melhorar a interpretação. Devido à estrutura inconsistente da base, alguns registros não correspondiam aos códigos padrão (0, 1, 3), sendo então alocados na categoria “Outros”.

**- Tratamento de NaN e Outliers (Normalização)**

**NaN (Valores Ausentes):** Após uma análise sobre os NaN, nós identificamos uma única linha com valores NaN em nossa base. Devido à baixa ocorrência, optamos por excluí-la, evitando impactos desnecessários na integridade do conjunto de dados.

**Outliers:** Para garantir a eficácia do modelo de IA, fizemos cinco técnicas de normalização para lidar com outliers:

- Capping: Limita valores extremos ao percentil máximo permitido.

- IQR (Interquartil Range): Exclui valores fora do intervalo interquartil.

- Log Transform: Reduz a magnitude dos valores altos, suavizando os outliers.

- Sqrt Transform: Aplica raiz quadrada aos dados para reduzir a influência de valores extremos.

Dos métodos testados, a normalização por log demonstrou os melhores resultados para o modelo, como será abordado na seção de análise de dados.

Para visualizar as estatísticas antes e depois da remoção dos outliers e facilitar a plotagem de gráficos, utilizamos a base com o método Capping, que corta os outliers e fornece uma perspectiva mais clara das distribuições.

- Os gráficos se encontram no mesmo notebook jupyter citado anteriormente.

Resumo estatístico do capital social pós normalização/corte de outliers e tipos dos dados:

UF object

Porte Empresa object

Capital Social float64

municipios object

cnaes object

Natureza Juridica object

ano inicio\_ativ int64

mes inicio\_ativ int64

dia inicio\_ativ int64

Status Operacional object

**Resumo estatístico capital social:**

Média: A média diminuiu para 4264.67, indicando que a distribuição geral foi afetada pela retirada dos valores extremos, comparado a mediana a média se manteve bem alta, mas agora com valores realista para o contexto de investimentos das MEIs e MEs.

**Mediana e Moda:**

Ambas permanecem em 5.0, mostrando que a concentração de valores na base continua em torno de números baixos, o que sugere que a maioria das empresas está nos níveis inferiores de "Capital Social."

**Desvio Padrão e Variância:**

O desvio padrão (35093.38) e a variância (1,231 bilhões) são ainda bastante altos, sinalizando uma variabilidade significativa, embora essa dispersão tenha sido um pouco reduzida em comparação ao estado inicial. Isso aponta para uma leve contenção de valores muito elevados.

**Amplitude e Máximo:**

A amplitude agora é 400,000, com o valor máximo ajustado também para 400,000, indicando que a remoção de outliers ajudou a restringir valores excessivamente altos, mantendo a base mais equilibrada.

**Quartis e IQR:**

O 1º quartil (Q1) é 1.0 e o 3º quartil (Q3) é 5.0, com a amplitude interquartílica (IQR) em 4.0, permanecendo praticamente inalterados. Esses valores sugerem que, no centro da distribuição, os dados continuam concentrados em torno de valores menores.

**Curtose e Assimetria:**

Embora a curtose (103.86) e a assimetria (9.95) ainda estejam elevadas, elas foram reduzidas pela limpeza, indicando uma distribuição menos concentrada nos valores extremos.

**Código da Limpeza completa:**

- O código da limpeza dos dados está no arquivo jupyter anexado ou no link do repositório:

<Link do repositório>

Nome arquivo jupyter:

DataCleaning.ipynb

Para plotagem dos dados limpos, monte um dashboard, de acordo com os requisitos da disciplina Business Intelligence (aproveite o que for desenvolvido para essa disciplina).

Como mencionado no tópico anterior, devido ao tamanho da base, o dashboard para a plotagem dos dados limpos também foi desenvolvido em Python, garantindo a eficiência na análise. Os gráficos e conclusões referentes à etapa de limpeza dos dados estão disponíveis no arquivo anexado:

*Graficos\_base\_gov.ipynb.*

# Condicionamento para alimentar o modelo de ML

O condicionamento de dados será realizado antes de rodar o modelo de ML. Também será feito após rodar o modelo de ML e for constatado que é necessário algum ajuste para reprocessamento.

Este item deverá conter uma subseção para cada etapa de condicionamento dos dados:

Para preparar os dados antes de alimentar o modelo de ML, realizamos um condicionamento estratégico para garantir a qualidade e a eficiência dos dados. Esse processo é dividido em etapas para garantir que o modelo receba dados otimizados, minimizando ruídos e maximizando a relevância das variáveis.

## Condicionamento inicial

* Normalização, padronização
* Redução de dimensionalidade
* Escolha de uma variável dentro de um conjunto de variáveis com alta correlação  
  etc.
* Avaliação dos dados preparados: Dashboard de Business Intelligence.

**- Tratamento de Dados Ausentes e Outliers**

**NaN:** Identificamos uma única linha com valores ausentes (NaN) e optamos por removê-la para não introduzir ruídos no modelo.

**Outliers:** Testamos cinco técnicas de normalização para suavizar os efeitos dos outliers nos nossos dados, sendo eles: capping, IQR (Intervalo Interquartil), log e sqrt. Eles ajudaram a preservar a base o mais informativa possível. Entre essas, a técnica de transformação logarítmica apresentou melhor desempenho, conforme avaliado posteriormente no modelo. A técnica de capping foi escolhida para visualizações, pois corta diretamente os valores atípicos, permitindo análise sem influência excessiva desses pontos.

**- Conversão de Variáveis Categóricas**

**Variáveis Categóricas para Numéricas:** As variáveis qualitativas foram convertidas para numéricas usando OrdinalEncoder, mantendo consistência na estrutura dos dados. Especificamente, colunas como Situação Cadastral foram mapeadas de código para uma representação textual para facilitar a compreensão e a visualização.

Reagrupamento de Classes: Unificamos categorias com semelhanças significativas. Por exemplo, as classes de situação cadastral foram reorganizadas em grupos como “Ativa”, “Irregular” (para classes Suspensa e Inativa), e “Encerrada” (para classes Baixada e Nula). Pois, com menos classes o modelo performaria melhor

**- Padronização**

Escalonamento e Padronização dos Dados: Utilizamos StandardScaler para padronizar a escala das variáveis contínuas, especialmente importante para o KNN, que é sensível a variações de escala.

Esse conjunto de etapas foi realizado para minimizar a interferência de valores discrepantes e para melhorar a representatividade dos dados, garantindo que o modelo de ML tenha uma base otimizada e de alta qualidade para seu aprendizado.

**ANÁLISE DE DADOS (MÓDOLO)**

# Definição dos objetivos e das classes

Falando especificamente sobre o modelo do nosso app, o objetivo central é prever o *status operacional* de MEIs (Microempreendedores Individuais) e MEs (Microempresas) por meio de um modelo de machine learning, com a finalidade de apoiar decisões estratégicas e otimizar a gestão de negócios desses segmentos. Este insight é crucial para gestores e órgãos interessados em identificar rapidamente a situação cadastral das empresas e planejar ações que promovam estabilidade e crescimento no setor. Além de um dado importantíssimo para nós da BiMo sabermos como guiar a MEI pelo melhor caminho possível.

## Objetivos Específicos:

A previsão do *status operacional* permite responder a perguntas-chave para o público-alvo e para as entidades reguladoras, incluindo:

* **Qual é o futuro mais provável da MEI/ME, baseado no Status Operacional da empresa cadastrada?** Saber se uma empresa está ativa, suspensa ou encerrada ajuda a identificar padrões de atividade e a tomar decisões mais informadas para cada situação.
* **Quais fatores influenciam o status operacional?** Entender as variáveis que impactam diretamente a situação cadastral pode auxiliar no desenvolvimento de estratégias de suporte para empresas que estão em risco de suspensão ou encerramento.
* **Como prever a transição entre os estados de atividade?** Detectar possíveis suspensões ou encerramentos com antecedência pode ser um recurso valioso para intervenções proativas. O que é extremamente importante para nosso contexto, já que a BiMo é uma rede de suporte para MEIs e Mês.

## Listar e descrever a(s) resposta(s) (y) e as classes que cada uma dessas respostas vai prever.

No nosso modelo, teremos 3 classes na coluna y (Status Operacional) e essas 3 classes são:

**- Ativa:**

Esta classe indica que a empresa está em operação, cumprindo com suas obrigações legais e regulatórias. Empresas nessa condição possuem autorização para exercer suas atividades comerciais.

**- Suspensa (Junção de Inapta e Suspensa):**

A classe suspensa é atribuída a empresas que interromperam temporariamente suas atividades. Esse status pode ser consequência de diferentes fatores, como problemas financeiros ou atrasos em conformidades fiscais.

**- Encerrada (Junção de Nula e Baixada):**

Empresas que se enquadram nesta classe foram formalmente fechadas ou tiveram seu registro cancelado. Esse status reflete o encerramento definitivo das atividades e marca a saída da empresa do mercado.

--- Como descrito anteriormente e como destacado entre os parênteses houve uma junção de classes na nossa base para melhor desempenho do modelo, essa junção foi brevemente explicada na parte de EAD do relatório e irá ser mais aprofundada nos tópicos a frente. ---

# Definição dos Modelos Mais Adequados para Analisar os Dados

Para a análise dos dados das MEIs e MEs e a previsão de seu *status operacional*, escolhemos modelos de classificação que apresentam uma combinação de melhor interpretação e precisão. Os modelos selecionados foram o **K-Nearest Neighbors (KNN)**, a **Árvore de Decisão** (*gini* e *entropy*) e o **Naive Bayes**. Abaixo, detalhamos os critérios para a escolha desses modelos e sua adequação para o problema proposto.

1. **Natureza dos Dados**
   * Considerando que os dados incluem variáveis categóricas e contínuas, os modelos escolhidos são flexíveis para lidar com diferentes tipos de dados após técnicas de codificação e padronização.
   * Como o objetivo é prever uma variável com três classes (*Ativa, Suspensa e Encerrada*), é necessário que o modelo tenha suporte para classificação multiclasse, característica que todos os modelos escolhidos atendem.
2. **Complexidade dos Modelos**
   * Optamos por modelos com complexidade baixa a média, para balancear melhor interpretação e capacidade de generalização. A Árvore de Decisão é um modelo simples e fácil de interpretar, permitindo entender as condições de decisão associadas a cada status.
   * **Naive Bayes** e **KNN** também são modelos de baixa complexidade que podem ser altamente eficazes para padrões claros entre as classes e que, ao mesmo tempo, são facilmente interpretáveis.
3. **Objetivos de Classificação**
   * Como se trata de um problema multiclasse, o KNN, Naive Bayes e a Árvore de Decisão são modelos que se adaptam bem a essa tarefa, oferecendo alternativas para explorar diferentes abordagens: desde a proximidade de características com o KNN, a probabilidade condicional com o Naive Bayes, até a segmentação em regras de decisão com a Árvore de Decisão.

# Visão Geral dos Modelos Considerados

* **K-Nearest Neighbors (KNN)**: Ideal para capturar semelhanças entre empresas com status operacional similar. Utilizando a proximidade entre as observações, o KNN consegue identificar padrões com base na vizinhança dos dados, podendo ser eficaz para identificar status semelhantes.
* **Árvore de Decisão (Gini e Entropy)**: As árvores de decisão são fáceis de interpretar e permitem entender como cada variável influencia o status operacional. Utilizar os critérios de *gini* e *entropy* permite explorar diferentes formas de segmentação dos dados, melhorando a precisão e a capacidade de generalização.
* **Naive Bayes**: Este modelo baseia-se em probabilidades condicionalmente independentes e é bem adequado para problemas de classificação rápida, mesmo com pequenas amostras. Com o Naive Bayes, podemos verificar a influência independente de cada variável sobre o status, o que facilita a interpretação e pode ser vantajoso para a classificação multiclasse.

# Critérios de Seleção dos Modelos

1. **Precisão e Robustez:** Modelos que ofereçam uma classificação confiável e sejam robustos para dados diversificados.
2. **Interpretação e Simplicidade:** A facilidade de interpretação é um fator muito importante, visto que precisamos entender as colunas que mais influênciam sobre o status operacional.
3. **Capacidade de Generalização para Classes Múltiplas:** Como o problema é multiclasse, todos os modelos escolhidos têm bom desempenho para esse tipo de tarefa e fornecem diferentes perspectivas para identificar o status de cada empresa.

# Descrição dos modelos selecionados

## Modelo 1: K-Nearest Neighbors (KNN)

O K-Nearest Neighbors (KNN) é um modelo baseado em instâncias que classifica uma amostra com base na similaridade entre os dados. Ao prever o status operacional de uma empresa, o KNN identifica os “vizinhos” mais próximos em termos de características e atribui a classificação da maioria desses vizinhos à amostra em análise.

**Critérios de Seleção do KNN:**

* **Precisão e Robustez:** O KNN tende a ter um bom desempenho em conjuntos de dados com grupos bem definidos e é sensível a padrões de proximidade, o que pode ser útil ao identificar empresas com características semelhantes em determinado status.
* **Interpretação e Explicabilidade:** O modelo é simples de interpretar, uma vez que a decisão de classificação se baseia na proximidade direta com outros dados no espaço de características.
* **Desempenho Computacional:** Embora o KNN seja computacionalmente intensivo em conjuntos de dados grandes, é eficaz para um volume moderado de dados. Para lidar com a complexidade, ajustes no número de vizinhos (k) podem ser feitos.
* **Capacidade de Generalização:** A capacidade de generalização do KNN depende fortemente da escolha de k e das características dos dados. A seleção de um valor apropriado de k ajuda a balancear entre o ajuste excessivo e a subutilização dos dados, aumentando a precisão nos dados de teste.

## Modelo 2: Árvore de Decisão (Gini e Entropy)

A **Árvore de Decisão** é um modelo de aprendizado supervisionado que cria uma estrutura hierárquica para tomada de decisões. Para classificar o status operacional de uma empresa, o modelo segue uma série de condições (nós) até atingir uma folha (classe). Foram escolhidos dois critérios de divisão: **Gini** e **Entropy**, ambos eficazes na criação de divisões informativas para variáveis categóricas e contínuas.

**Critérios de Seleção da Árvore de Decisão:**

* **Precisão e Robustez:** A árvore de decisão é robusta para diferentes tipos de dados e tende a gerar previsões precisas com base em divisões informativas. Os critérios de Gini e Entropy ajudam a identificar os melhores pontos de divisão.
* **Interpretação e Explicabilidade:** Este modelo é altamente interpretável e visualmente intuitivo, facilitando a compreensão de como cada característica impacta o status operacional.
* **Desempenho Computacional:** A árvore de decisão é geralmente eficiente e rápida de treinar, especialmente com critérios de divisão rápidos como Gini e Entropy, o que possibilita a implementação em tempo hábil.
* **Capacidade de Generalização:** Com podas adequadas, a árvore de decisão evita o ajuste excessivo e se generaliza bem para dados de teste. As configurações para limitar a profundidade ou o número mínimo de amostras por folha ajudam a melhorar a generalização.

## Modelo 3: Naive Bayes

O **Naive Bayes** é um modelo probabilístico que assume independência condicional entre as variáveis. Ele calcula a probabilidade de uma amostra pertencer a uma classe com base na probabilidade de cada característica, sendo especialmente eficiente para classificações rápidas e baseadas em probabilidades.

**Critérios de Seleção do Naive Bayes:**

* **Precisão e Robustez:** Naive Bayes é conhecido por sua precisão em problemas com variáveis independentes e, apesar da suposição de independência, o modelo é robusto para classificação multiclasse, como é o caso do status operacional.
* **Interpretação e Explicabilidade:** As previsões probabilísticas são fáceis de entender, pois o modelo permite avaliar diretamente a probabilidade de cada classe.
* **Desempenho Computacional:** Naive Bayes é muito eficiente computacionalmente, pois calcula as probabilidades de forma rápida, o que é vantajoso em conjuntos de dados grandes.
* **Capacidade de Generalização:** Devido à sua abordagem probabilística, o modelo tende a generalizar bem para novos dados, especialmente quando há padrões claros nas probabilidades das classes.

# Aplicação dos Modelos Selecionados

## Divisão dos Dados

* **Divisão entre Treino e Teste**: A divisão dos dados foi realizada utilizando o método train\_test\_split, com 75% dos dados alocados para o conjunto de treinamento e 25% para o conjunto de teste. Esta abordagem ajuda a garantir que o modelo seja treinado em uma quantidade substancial de dados e testado de forma independente em um conjunto separado, proporcionando uma avaliação mais precisa de seu desempenho em dados não vistos.
* **Validação Cruzada**: Para assegurar uma avaliação ainda mais robusta, foi aplicada a técnica de validação cruzada K-Fold, com 4 divisões (n\_splits=4). O K-Fold cria 4 divisões dos dados, onde cada uma das partes é utilizada como conjunto de teste uma vez, enquanto as outras três servem como treino. A média dos resultados das 4 iterações proporciona uma avaliação mais equilibrada do modelo, mitigando o risco de viés de partição.

## Treinamento dos Modelos

* **Árvores de Decisão (Critérios de Gini e Entropia)**: Foram utilizados dois classificadores de árvore de decisão, um configurado com o critério *Gini* (classificador\_gini) e outro com o critério *Entropia* (classificador\_entropy). Ambos foram configurados com parâmetros de máxima profundidade (max\_depth=8), número mínimo de amostras para divisão (min\_samples\_split=25) e número mínimo de amostras em cada folha (min\_samples\_leaf=25). Esses parâmetros foram ajustados para evitar overfitting e otimizar a capacidade de generalização das árvores.
* **Naive Bayes**: O classificador GaussianNB foi treinado como um modelo probabilístico para classificação. Este modelo é adequado para dados contínuos e categóricos convertidos, como os utilizados após o pré-processamento. Sua simplicidade e eficiência fazem dele uma escolha ideal para uma classificação rápida em problemas multiclasse.
* **K-Nearest Neighbors (KNN)**: Também foi treinado um modelo de vizinhos mais próximos (KNeighborsClassifier). Esse modelo considera a proximidade entre as instâncias para fazer suas previsões, sendo configurado sem hiperparâmetros específicos, já que ele foi aplicado na configuração padrão para avaliar seu desempenho em comparação aos demais modelos.

## Métricas de Avaliação

* **Métricas Utilizadas**:
  + **Acurácia**: Avalia a porcentagem de previsões corretas em relação ao total de previsões. É uma métrica útil para verificar a precisão geral do modelo.
  + **Precisão(precision)**: Mede a proporção de previsões corretas entre todas as previsões positivas, sendo crucial para minimizar falsos positivos em cada classe.
  + **Recall**: Avalia a capacidade do modelo de identificar corretamente todas as instâncias de uma classe positiva entre as amostras verdadeiras, essencial para casos em que é importante detectar todas as ocorrências de uma classe.
  + **F1-Score**: Calcula a média harmônica entre precisão e recall, proporcionando uma visão equilibrada entre ambos, especialmente útil em situações em que as classes estão desbalanceadas.
* **Relatório de Classificação**: O desempenho de cada modelo foi analisado através do classification\_report para o conjunto de treino e teste. Este relatório oferece uma visão detalhada das métricas por classe, permitindo verificar o desempenho individual em "Ativa", "Suspensa" e "Encerrada."

# Análise dos Resultados

## Modelo 1: Árvore de Decisão com Critério Gini

* **Métricas de Avaliação**:
  + *Acurácia*: 79% nos dados de teste e 79% nos dados de treino.
  + *Precisão e Recall*: A classe “NULA” obteve alta precisão (0.87) e recall (0.89), enquanto as demais classes, como “ATIVA” e “SUSPENSA,” apresentaram precisão e recall muito baixos (0.00).
  + *F1-Score*: A média ponderada foi de 0.78, refletindo um bom desempenho para a classe majoritária “NULA” e dificuldades nas outras classes.
* **Discussão dos Resultados**:
  + **Equilíbrio entre Precisão e Recall**: O modelo apresentou alto equilíbrio para a classe “NULA”, mas baixo equilíbrio para as demais classes. Isso indica que ele está bem ajustado para a classe majoritária, mas não identifica adequadamente as minoritárias.
  + **Classes Desbalanceadas**: Como esperado, a classe “NULA” domina as previsões, enquanto as classes menos frequentes (“ATIVA” e “SUSPENSA”) têm desempenho quase nulo, mostrando que o modelo não consegue aprender corretamente essas classes menos representadas.
  + **Erro e Overfitting**: Com a mesma acurácia em treino e teste (79%), o modelo não mostra sinais claros de overfitting, mas tem dificuldade em generalizar para as classes minoritárias, o que reflete uma limitação na sua capacidade de captura de padrões minoritários.

## Modelo 2: Árvore de Decisão com Critério Entropia

* **Métricas de Avaliação**:
  + *Acurácia*: 79% nos dados de teste e 79% nos dados de treino.
  + *Precisão e Recall*: Alta precisão (0.85) e recall (0.91) para a classe “NULA”; no entanto, o recall é zero para “ATIVA” e “SUSPENSA”.
  + *F1-Score*: A média ponderada foi 0.77, com maior ênfase na classe “NULA.”
* **Discussão dos Resultados**:
  + **Equilíbrio entre Precisão e Recall**: Semelhante ao modelo Gini, o critério de Entropia apresentou bom equilíbrio entre precisão e recall para a classe “NULA,” mas baixo para as outras, especialmente “INAPTA” e “BAIXADA”.
  + **Classes Desbalanceadas**: Novamente, o desempenho é muito influenciado pela classe “NULA,” mostrando uma tendência de enviesamento para a classe majoritária. A dificuldade em classificar as demais classes sugere que o modelo prioriza a informação da classe dominante.
  + **Erro e Overfitting**: Com métricas consistentes entre treino e teste, o modelo não sofre de overfitting, mas também mostra baixa capacidade de previsão para classes menores.

## Modelo 3: Naive Bayes

* **Métricas de Avaliação**:
  + *Acurácia*: Apenas 13% nos dados de teste e 15% nos dados de treino.
  + *Precisão e Recall*: A classe “NULA” tem precisão alta (1.00) no treino, mas recall baixo (0.00), e “INAPTA” tem recall muito alto (0.99), embora seu f1-score seja baixo.
  + *F1-Score*: A média ponderada do f1-score foi extremamente baixa (0.04), refletindo uma inconsistência geral.
* **Discussão dos Resultados**:
  + **Equilíbrio entre Precisão e Recall**: O modelo apresentou um desbalanceamento acentuado entre precisão e recall, indicando problemas em prever adequadamente as classes e possíveis erros nos parâmetros do modelo.
  + **Classes Desbalanceadas**: Naive Bayes não conseguiu capturar adequadamente as classes, apresentando recall quase nulo para as classes minoritárias, o que comprometeu o seu desempenho geral.
  + **Erro e Overfitting**: O modelo apresenta baixo desempenho em ambos os conjuntos, indicando falta de adaptação aos dados, possivelmente devido à suposição de independência entre as variáveis, que não é adequada para este conjunto.

## Modelo 4: K-Nearest Neighbors (KNN)

* **Métricas de Avaliação**:
  + *Acurácia*: 73% nos dados de teste e 78% nos dados de treino.
  + *Precisão e Recall*: Alta precisão (0.78) e recall (0.93) para a classe “NULA”, mas baixo recall para as classes “ATIVA” e “SUSPENSA.”
  + *F1-Score*: Média ponderada de 0.69, indicando que o modelo capturou parcialmente a classe “INAPTA” e “BAIXADA”, mas teve dificuldades nas demais.
* **Discussão dos Resultados**:
  + **Equilíbrio entre Precisão e Recall**: Embora o modelo apresente um bom equilíbrio para a classe “NULA,” as outras classes sofrem de baixos índices de recall, o que diminui a eficácia do modelo em capturar adequadamente as classes minoritárias.
  + **Classes Desbalanceadas**: KNN teve dificuldades em geral para capturar as classes menos frequentes. Isso se deve à sensibilidade do KNN à distribuição de classes, especialmente em casos de desbalanceamento.
  + **Erro e Overfitting**: A diferença entre treino e teste é pequena, mostrando pouca propensão a overfitting, mas ainda assim o modelo apresenta limitações em classes minoritárias.

## Comparação Direta

1. **Acurácia**: Os modelos de árvore de decisão (Gini e Entropia) foram os que apresentaram melhor acurácia (79%), seguidos pelo KNN (73%).
2. **Precisão e Recall**: As árvores de decisão apresentaram maior precisão e recall na classe “NULA”, mas nenhum modelo conseguiu bons resultados nas classes “ATIVA” e “SUSPENSA”.
3. **Classes Desbalanceadas**: Todos os modelos sofreram com o desbalanceamento, mas o Naive Bayes foi o que apresentou o pior desempenho global.

## Forças e Fraquezas

* **Árvores de Decisão (Gini e Entropia)**:
  + *Forças*: Boa acurácia e desempenho consistente na classe “NULA.”
  + *Fraquezas*: Fraca capacidade de generalizar para classes minoritárias, levando a um desempenho baixo em classes desbalanceadas.
* **Naive Bayes**:
  + *Forças*: Processamento rápido e simples.
  + *Fraquezas*: Desempenho muito ruim em classes desbalanceadas, acurácia geral baixa, e inadequado para a complexidade das variáveis.
* **K-Nearest Neighbors (KNN)**:
  + *Forças*: Bom desempenho em termos de precisão para a classe “NULA”.
  + *Fraquezas*: Baixa capacidade de identificar classes minoritárias e sensibilidade ao desbalanceamento.

A análise destaca que as árvores de decisão apresentam o melhor desempenho geral para este problema, sendo mais robustas e com menor desvio entre as classes majoritárias e minoritárias. Contudo, nenhuma técnica foi capaz de mitigar totalmente o problema do desbalanceamento, o que sugere a necessidade de técnicas adicionais para melhorar o desempenho nas classes menos frequentes.

# Ajustes Necessários

#### Identificação de Problemas

1. **Problemas de Desempenho**:
   * Durante a avaliação, os modelos apresentaram algumas dificuldades com classes específicas, especialmente as classes minoritárias, como "Suspensa" e "Inapta" (agora reclassificadas como “Irregular”).
   * Observou-se baixa precisão e recall para essas classes, indicando que os modelos tendiam a prever mais frequentemente a classe “Ativa” ou "Encerrada", que são majoritárias, resultando em uma classificação desbalanceada.
2. **Dados Desbalanceados**:
   * A análise inicial indicou que o desbalanceamento entre as classes estava influenciando o desempenho, especialmente para o modelo GaussianNB, que é sensível a distribuições desiguais de classes. Modelos como KNN e Árvore de Decisão também foram afetados, pois esses algoritmos têm desempenho reduzido quando as classes não estão igualmente representadas.

#### Mudanças na Base de Dados

1. **Junção de Classes**:
   * Como parte do ajuste, foi realizada uma fusão de classes para simplificar o problema e reduzir o desbalanceamento extremo. As classes “Nula” e “Baixada” foram combinadas em “Encerrada,” enquanto “Suspensa” e “Inapta” foram combinadas como “Irregular”. Essa mudança diminuiu a variabilidade das classes e aumentou o número de amostras por classe, o que melhorou o desempenho geral dos modelos.
2. **Balanceamento de Classes**:
   * Técnicas de balanceamento, como SMOTE, foram consideradas para algumas bases, especialmente na construção dos modelos que apresentaram desempenho inferior, como GaussianNB. Contudo, essa abordagem foi deixada de lado, pois tentamos aplicá-la, porém surpreendentemente as métricas pioraram levemente, ou se mantiveram iguais, já que a fusão das classes ajudou a reduzir parte do problema de desbalanceamento.

#### Pré-Processamento

1. **Limpeza de Dados**:
   * Os dados foram revisados para identificar e tratar outliers. Em algumas variações de bases, como Base\_capping e Base\_iqr, foram utilizados métodos como capping e IQR para tratar valores extremos. Essas modificações visaram reduzir o impacto de outliers, especialmente em variáveis de natureza contínua, como o capital social.
2. **Transformação de Dados**:
   * Foi aplicada uma normalização usando StandardScaler para garantir que os modelos, especialmente o KNN e o Naive Bayes, pudessem interpretar melhor os dados sem serem afetados por escalas diferentes entre variáveis.
   * Diversas versões da base foram testadas, incluindo transformações logarítmicas e de raiz quadrada, buscando identificar quais métodos resultavam em melhor desempenho, especialmente para o modelo GaussianNB.
3. **Engenharia de Características**:
   * Foi usada a técnica SelectKBest para reduzir o número de colunas de 16 para no máximo 10. Isso foi útil para identificar as variáveis mais relevantes, facilitar a interpretação e reduzir a complexidade computacional.
   * Após a seleção de características, a técnica de PCA foi aplicada para reduzir a dimensionalidade e garantir que o máximo de variância dos dados fosse preservado, especialmente para melhorar o desempenho de modelos sensíveis à dimensionalidade, como KNN e Naive Bayes.

#### Ajustes nos Modelos

1. **Ajuste de Hiperparâmetros**:
   * Uma busca em grade (GridSearchCV) foi utilizada para ajustar hiperparâmetros de cada modelo. Por exemplo, para o DecisionTreeClassifier, foram testados diferentes critérios de divisão (gini e entropia), profundidades máximas, e mínimos de amostras por folha e por divisão. Isso permitiu encontrar a melhor combinação de parâmetros para otimizar a acurácia.
   * Para o modelo GaussianNB, o hiperparâmetro var\_smoothing foi ajustado em uma escala logarítmica para lidar com o problema de variância das features.
2. **Validação Cruzada**:
   * A validação cruzada foi aplicada a cada modelo para avaliar seu desempenho em diferentes partições da base e evitar overfitting. Com o cross\_validate, foi possível calcular a média das métricas de acurácia, precisão, recall e F1-score, o que resultou em uma avaliação mais robusta e na escolha do melhor modelo para cada variação da base.

#### Impactos observados nos Resultados

1. **Melhora na Generalização e Desempenho**:
   * A reclassificação das classes, junto com o uso de SelectKBest e PCA, resultou em uma melhora de aproximadamente 10% nas métricas de acurácia e F1-score, especialmente para os modelos DecisionTreeClassifier e KNN. Essa simplificação das classes permitiu que o modelo conseguisse generalizar melhor e equilibrar o recall entre as classes.
2. **Redução de Problemas de Overfitting**:
   * A aplicação da validação cruzada e o ajuste de hiperparâmetros reduziram o overfitting em vários modelos. Notou-se uma menor discrepância entre as métricas de treino e teste, sugerindo que os modelos estavam mais equilibrados para dados desconhecidos.

# Impacto das Modificações nos Modelos

## Modelo 1: GaussianNB

**Discussão dos Resultados:** Após os ajustes na base e os refinamentos dos hiperparâmetros, o modelo GaussianNB apresentou melhorias, principalmente em termos de **acurácia** e **média de precisão**. Isso ocorreu devido à utilização de técnicas de redução de dimensionalidade (SelectKBest e PCA), que possibilitaram ao GaussianNB trabalhar com um conjunto de variáveis mais relevante, minimizando o ruído e as variáveis redundantes que poderiam afetar o desempenho.

A modificação mais impactante foi a aplicação de transformação logarítmica e de raiz quadrada no Capital Social, que melhorou a média de precisão para 0.44 e a média de F1 para 0.34. Esses ajustes foram fundamentais, considerando que o GaussianNB tem dificuldade com dados que não são normalmente distribuídos. A fusão de classes menos frequentes também foi essencial para melhorar a capacidade do modelo de diferenciar corretamente entre as categorias “Encerrada” e “Irregular,” resultando em uma ligeira elevação no **recall médio** para 0.36.

## Modelo 2: KNN

**Discussão dos Resultados:** Para o KNN, as transformações de variáveis e a redução de dimensionalidade resultaram em ganhos significativos de desempenho. Com as mudanças, o modelo alcançou uma **acurácia de até 0.77**, e o ajuste de hiperparâmetros foi fundamental para otimizar o número de vizinhos e o valor de componentes principais no PCA, o que elevou a média de **F1 para 0.6**.

A técnica de oversampling ou reclassificação de classes para balanceamento trouxe uma melhora perceptível na média de **recall**, especialmente para classes anteriormente minoritárias. A normalização das variáveis com StandardScaler contribuiu para uniformizar as distâncias euclidianas, algo importante para o KNN, que depende da distância entre pontos para realizar as previsões.

## Modelo 3: DecisionTreeClassifier

**Discussão dos Resultados:** O DecisionTreeClassifier mostrou-se particularmente responsivo aos ajustes na base e à reclassificação das classes. Após os ajustes, a árvore de decisão obteve um aumento na **acurácia média para 0.76** e um aumento notável no **F1 médio para 0.58**. Esses resultados refletem a capacidade do modelo de capturar relações complexas na base após a aplicação de SelectKBest, que permitiu reduzir o número de variáveis menos significativas e tornar a árvore menos propensa ao overfitting.

Além disso, a aplicação de PCA após a seleção de variáveis ajudou o modelo a lidar com a variância reduzida nas classes “Encerrada” e “Irregular,” resultando em melhorias no **recall médio**. A seleção de uma profundidade máxima adequada e um número mínimo de amostras por folha durante o ajuste de hiperparâmetros também ajudou a manter o balanceamento entre precisão e recall, especialmente nas classes minoritárias, o que contribuiu para uma maior estabilidade no desempenho geral do modelo.

Essas modificações resultaram em um aumento geral das métricas para todos os modelos, com melhorias específicas em áreas onde cada um deles enfrentava mais dificuldades.

## Comparação Entre Modelos

#### Comparação dos Resultados das Métricas de Avaliação

Os três modelos, GaussianNB, KNN e DecisionTreeClassifier, foram avaliados em termos de acurácia, precisão, recall e F1-score, com as seguintes observações principais:

1. **GaussianNB**:
   * **Força**: O GaussianNB teve um desempenho estável e uma implementação rápida. Após os ajustes na base e a aplicação de redução de dimensionalidade, ele alcançou uma **acurácia de aproximadamente 0.71** e uma **precisão média** de 0.44.
   * **Fraqueza**: Este modelo apresentou limitações ao lidar com classes desbalanceadas e obteve uma média de recall de apenas 0.36. Isso sugere uma dificuldade em identificar corretamente as instâncias de classes minoritárias, especialmente nas classes “Irregular” e “Encerrada”.
   * **Adequação**: O GaussianNB é adequado para problemas onde as classes são aproximadamente equilibradas e as variáveis seguem uma distribuição normal. Embora tenha apresentado uma boa precisão, ele não foi o melhor modelo para capturar as nuances dos dados, especialmente em classes menos representadas.
2. **KNN**:
   * **Força**: O KNN demonstrou ser robusto em relação às variações nas variáveis. Após os ajustes de hiperparâmetros e a aplicação de SelectKBest e PCA, ele obteve uma **acurácia máxima de 0.77** e uma **F1-score médio de 0.6**. Isso indica uma boa capacidade de classificação em geral, principalmente após a normalização dos dados, o que é essencial para um modelo baseado em distâncias.
   * **Fraqueza**: A principal limitação do KNN foi o tempo de execução e a sensibilidade ao desbalanceamento de classes. Além disso, seu desempenho em classes minoritárias ainda foi moderado, exigindo um número adequado de vizinhos para alcançar um bom equilíbrio entre precisão e recall.
   * **Adequação**: O KNN é uma boa escolha para dados em que a estrutura espacial é importante. Ele se beneficiou de uma base de dados bem pré-processada e balanceada, tornando-se uma alternativa viável, embora sua execução seja mais lenta e tenha limitações para escalabilidade.
3. **DecisionTreeClassifier**:
   * **Força**: O DecisionTreeClassifier destacou-se com a melhor média de **recall (0.56)** e uma acurácia elevada, alcançando **até 0.76** após os ajustes de hiperparâmetros e a reclassificação de classes. A árvore de decisão conseguiu lidar bem com o desbalanceamento, graças à capacidade de se adaptar aos dados de treinamento e ajustar suas profundidades para evitar overfitting.
   * **Fraqueza**: A árvore de decisão é mais suscetível ao overfitting em bases complexas ou com muitas variáveis. Porém, a aplicação de PCA e SelectKBest ajudou a mitigar essa limitação.
   * **Adequação**: Este modelo mostrou-se altamente adequado para os dados, pois capturou melhor as relações complexas entre as variáveis e teve um desempenho equilibrado nas classes minoritárias. O DecisionTreeClassifier demonstrou a melhor capacidade de generalização entre as classes após ajustes específicos.

#### Escolha do Modelo Mais Adequado

O **modelo mais adequado para esses dados é o** DecisionTreeClassifier. Sua capacidade de lidar com classes desbalanceadas e complexidade nas variáveis foi comprovada através das métricas de recall e F1-score, que foram mais equilibradas em comparação com os outros modelos. Além disso, o DecisionTreeClassifier apresentou uma adaptabilidade significativa após ajustes de hiperparâmetros e seleção de variáveis, garantindo uma boa generalização e reduzindo o risco de overfitting.

Em resumo, o DecisionTreeClassifier é o modelo mais robusto para este contexto devido à sua capacidade de capturar interações não lineares, ajustabilidade por hiperparâmetros, e à sua vantagem em termos de precisão e recall nas classes relevantes.

# Conclusão

Esta análise revelou importantes insights sobre a capacidade preditiva de três modelos de classificação – GaussianNB, KNN e DecisionTreeClassifier – no contexto de previsão do status operacional das empresas. Durante o processo, foram aplicadas diversas técnicas para aprimorar o desempenho dos modelos, incluindo ajustes de hiperparâmetros, transformações e seleção de variáveis, além de métodos de redução de dimensionalidade. Esses ajustes permitiram observar nuances nos dados e melhorar o desempenho em diferentes métricas.

#### Principais Descobertas

* **Desempenho do DecisionTreeClassifier**: O DecisionTreeClassifier se destacou por seu desempenho consistente, atingindo uma **acurácia de 0.76** e um **F1-score médio de 0.58**. Este modelo foi particularmente eficaz na captura de padrões em classes minoritárias, como “Encerrada” e “Irregular”, após ajustes nos hiperparâmetros e balanceamento da base. A árvore de decisão foi capaz de lidar com a complexidade dos dados e ofereceu uma boa capacidade de generalização, o que o posiciona como o modelo mais adequado para a base atual.
* **KNN e Robustez Estrutural**: O KNN apresentou uma robustez estrutural, alcançando **acurácia de até 0.77** e **F1-score médio de 0.6** após normalização e seleção de variáveis. Esse modelo demonstrou ser bem-sucedido em capturar a distribuição espacial dos dados, embora tenha se mostrado sensível ao desbalanceamento de classes e tenha demandado uma execução mais lenta para lidar com o aumento de vizinhos. Apesar disso, o KNN manteve um bom equilíbrio entre precisão e recall.
* **Limitantes do GaussianNB**: O GaussianNB foi o que mais surpreendeu em termos de tempo de execução, sendo o modelo mais lento, o que pode impactar sua viabilidade prática em aplicações em tempo real ou de alta demanda. Ele alcançou uma acurácia média de 0.71, mas teve dificuldade em lidar com o desbalanceamento das classes, com uma média de recall de 0.36, revelando limitações em capturar classes minoritárias.

#### Impacto Potencial das Previsões

As previsões dos modelos podem trazer benefícios estratégicos para o projeto, ao auxiliar na identificação de empresas ativas e em potencial risco de irregularidade. A precisão dessas previsões permite otimizar recursos e minimizar riscos nas interações com diferentes empresas, contribuindo para decisões mais informadas e alinhadas com o objetivo do projeto.

#### Próximas Etapas

Para aprimorar ainda mais os resultados podemos utilizar técnicas mais avançadas como:

1. **Explorar Novas Variáveis e Interações**: O desenvolvimento de novas variáveis derivadas pode ajudar a capturar relações mais profundas nos dados, otimizando a precisão dos modelos.
2. **Considerar Modelos Ensemble**: O uso de técnicas como Random Forest e Gradient Boosting pode reforçar a capacidade preditiva dos modelos, proporcionando ganhos em desempenho.
3. **Monitoramento e Ajuste Contínuo**: A implementação de monitoramento constante permitirá avaliar o modelo em cenários reais e ajustá-lo conforme novos dados forem coletados, garantindo melhor desempenho a longo prazo.

### Conclusão Final

A análise demonstrou que o DecisionTreeClassifier é o modelo mais adequado para a base atual, equilibrando desempenho e tempo de execução. A aplicação de ajustes específicos e de novas recomendações poderá melhorar ainda mais o sistema de previsão, fortalecendo sua contribuição para o sucesso do projeto.

# RPA: descrição do funcionamento e código python

Foram produzidos 3 RPA’s, para automação de atualização e para transação de dados:

Funcionalidades dos 3 RPA’s:

* 🎲 Atualização de dados administrativos: Atualizaremos as informações no banco de dados do app com as informações que nosso Adm's adicionarem/atualizarem/excluírem no banco de dados deles.  
  Para as seguintes tabelas (tabelas deles -> nossas tabelas):
  + CategoriaProduto e CategoriaCurso -> Categoria
  + Plano -> Plano
  + Curso -> Curso
  + MidiaCurso -> Curso
* 📈 Atualização de visualização Dash local: Atualizaremos as visualizações e dados de um Dashboard que contém avaliações e informações valiosas para nosso projeto.
* 📊 Atualização de visualização Dash feira: Atualizaremos as visualizações e dados de um Dashboard que contém avaliações do dia da feira do interdisciplinar.

#### **Primeiro RPA (Forms) -** [**Rpa\_BI.py**](https://github.com/BiMODados/Rpas_Interdisciplinar/blob/PROD/Rpa_BI.py)

**Objetivo**

Automatiza a atualização das informações em um dashboard de PowerBI com análises gráficas da pesquisa realizada via Forms.

**Processo**

1. O RPA abre o site que hospeda o modelo semântico do dashboard de Forms.
2. Como o acesso é não autorizado, o RPA utiliza variáveis de ambiente (email e senha) para autenticação.
3. Após carregar as variáveis e a página, ele realiza o login, navegando pelo site e passando por várias etapas.
4. Uma vez autenticado, o RPA localiza e clica no botão de atualização.
5. O RPA retorna uma mensagem indicando o sucesso da operação ou, em caso de erro, interrompe o processo e registra logs no AWS Cloud EC2 para monitoramento.

**Bibliotecas Utilizadas**

* Selenium (webdriver, Service, ChromeDriverManager, By, WebDriverWait, EC)
* dotenv (load\_dotenv)

**Cloud: AWS EC2**

#### **Segundo RPA (Feira) -** [**Rpa\_BI\_Feira.py**](https://github.com/BiMODados/Rpas_Interdisciplinar/blob/PROD/Rpa_BI_Feira.py)

**Objetivo**

Este RPA atualiza o dashboard da feira, onde são registradas avaliações realizadas no evento interdisciplinar.

**Processo**

1. Abre o site do modelo semântico do dashboard da feira.
2. Usa variáveis de ambiente para login, já que o acesso é não autorizado.
3. Realiza o login e navega por diversas telas até acessar o painel desejado.
4. Uma vez autenticado, localiza e clica no botão de atualização.
5. Retorna uma mensagem de sucesso ou erro, com logs armazenados no AWS Cloud EC2.

**Bibliotecas Utilizadas**

* Selenium (webdriver, Service, ChromeDriverManager, By, WebDriverWait, EC)
* dotenv (load\_dotenv)

**Cloud: AWS EC2**

### **Script de Modelagem de Dados -** [**Script\_Norm.py**](https://github.com/BiMODados/Rpas_Interdisciplinar/blob/PROD/Script_Norm.py)

**Objetivo**

Automatiza a atualização do banco de dados do aplicativo com as modificações realizadas pelos administradores no banco de dados administrativo. A atualização se aplica às seguintes tabelas:

* CategoriaProduto e CategoriaCurso ➔ Categoria
* Plano ➔ Plano
* Curso ➔ Curso
* MidiaCurso ➔ Curso

**Processo**

Foram criadas as seguintes funções para a automação do processo de atualização:

* **transferir\_dados(tabela\_origem, tabela\_destino, tipo\_categoria, colunas)**

Esta função padroniza o processo de transferência de dados, com algumas ramificações específicas para melhor atender as tabelas.

* + **Parâmetros**:
    - tabela\_origem: Tabela do banco administrativo para extração de dados.
    - tabela\_destino: Tabela no banco do aplicativo para inserção dos dados extraídos.
    - tipo\_categoria: Identifica o tipo de categoria para ajustar o fluxo de inserção, utilizando um ID alfanumérico, se necessário.
    - colunas: Colunas que receberão os dados.
* **transferir\_dados\_geral()**

Esta função chama transferir\_dados e inclui um fluxo adicional para tratar a inserção de dados na tabela de cursos.

* **update\_data(tabela\_origem, tabela\_destino, tipo\_categoria, colunas)**

Atualiza ou exclui dados no banco com base no campo booleano bIsUpdated, que indica se houve alteração na linha. Os parâmetros são os mesmos de transferir\_dados.

Para monitoramento, foram adicionados logs que registram o funcionamento do script.

**Bibliotecas Utilizadas**

* psycopg2
* dotenv (load\_dotenv)

**Cloud: AWS EC2**

# Modelo no APP: descrição do funcionamento do modelo e explicação de como foi publicado no app

### **Aplicação do modelo:**

Para aplicar o modelo no App, optamos por serializar ele e subir uma api que faça requisições ao modelo e guarde os resultados e informações recebidas em nosso banco de dados para consultas futura.

**Funcionalidades**

* **Validação e recebimento de dados empresariais:** Recebe e valida informações essenciais sobre empresas, verificando campos obrigatórios e o formato dos dados.
* **Transformação, classificação e armazenamento**: Recebe as informações e classifica utilizando uma inteligência artificial. Após essa classificação insere os resultados e dados em nosso banco de dados.
* **Resposta detalhada em caso de erros:** A API retorna mensagens claras e estruturadas, facilitando a identificação e correção de problemas.
* **Campos suportados:** São validados campos como email, nome da empresa, unidade federativa, porte e natureza jurídica, além de detalhes de início de atividades.

**Campos Obrigatórios (I.A)**

Os seguintes campos devem estar presentes na requisição:

* **email:** Endereço de e-mail da empresa. (Formato: string)
* **nome\_empresa:** Nome da empresa (até 100 caracteres).
* **uf:** Unidade Federativa, aceitando valores específicos (ex. SP, RJ).
* **porte\_empresa:** Porte da empresa (Microempresa, Empresa de Pequeno Porte, Outros).
* **capital\_social:** Capital social em reais (formato: inteiro).
* **municipios:** Município de registro, sem acentos.
* **cnaes:** Classificação de atividades econômicas.
* **natureza\_juridica:** Natureza jurídica da empresa.
* **ano\_inicio\_ativ:** Ano de início das atividades (inteiro).
* **mes\_inicio\_ativ:** Mês de início das atividades (1-12).
* **dia\_inicio\_ativ:** Dia de início das atividades (1-31).

**Estrutura de Requisição**

* **Endpoint:** (POST) /getResponse/

**Exemplo de Body:**

{  
 "email": "[contato@exemplo.com](mailto:contato@exemplo.com)",  
 "nome\_empresa": "Exemplo LTDA",  
 "uf": "SP",  
 "porte\_empresa": "Micro Empresa",  
 "capital\_social": 50000,  
 "municipios": "SAO PAULO",  
 "cnaes": "Comércio varejista de artigos do vestuário e acessórios",  
 "natureza\_juridica": "Sociedade Empresária Limitada",  
 "ano\_inicio\_ativ": 2015,  
 "mes\_inicio\_ativ": 6,  
 "dia\_inicio\_ativ": 15  
}

Para implementar no App criamos um site front-end que consome essa api e cria um formulário para estas informações que será carregado por meio de uma webview no início do app para que possamos captar o máximo de usuários.

Links dos repositórios:  
<https://github.com/BiMODados/Rpas_Interdisciplinar>

<https://github.com/BiMODados/Api_AI_Inter>

Link do repositório com a criação do modelo e o Data Cleaning:

<https://github.com/BiMODados/AED_AD.git>