Análise de Dados – Exploratória e ML

Sumário

[Introdução 4](#_Toc181777302)

[Objetivo do aplicativo 4](#_Toc181777303)

[Objetivo da Análise Exploratória de Dados 4](#_Toc181777304)

[Objetivo da Análise de Dados (para Aprendizado de Máquina) 4](#_Toc181777305)

[Levantamento dos dados na análise exploratória 4](#_Toc181777306)

[Busca dos dados 4](#_Toc181777307)

[Justificativa de uso 4](#_Toc181777308)

[Descrição da base de dados de trabalho 5](#_Toc181777309)

[Limpeza dos dados 6](#_Toc181777310)

[Condicionamento para alimentar o modelo de ML 6](#_Toc181777311)

[Condicionamento inicial 7](#_Toc181777312)

[Definição dos objetivos e das classes 7](#_Toc181777313)

[Definição dos modelos mais adequados para analisar os dados 7](#_Toc181777314)

[Descrição dos modelos selecionados 7](#_Toc181777315)

[Modelo 1: Naive Bayes 7](#_Toc181777316)

[Modelo 2: KNN 8](#_Toc181777317)

[Modelo 3: Decision tree por ccp alphas 8](#_Toc181777318)

[Modelo 4: Decision tree com hyperparametros 8](#_Toc181777319)

[Modelo 5: SVC 8](#_Toc181777320)

[Aplicação dos modelos selecionados 8](#_Toc181777321)

[Análise dos Resultados 9](#_Toc181777322)

[Modelo 1: Naive Bayes 9](#_Toc181777323)

[Modelo 2: KNN 10](#_Toc181777324)

[Modelo 3: Decision tree com CCP Alphas 12](#_Toc181777325)

[Modelo 4: Decision tree com Hyperparamentros 14](#_Toc181777326)

[Modelo 1: SVC 16](#_Toc181777327)

[Comparação Entre Modelos: 18](#_Toc181777328)

[Publicação do Modelo no APP 19](#_Toc181777329)

[Conclusão análise de dados 19](#_Toc181777330)

[Utilização do RPA no projeto 19](#_Toc181777331)

Análise de Dados para Tropicalias

MOC – Mosaico Cultural

Felipe Ricardo Ramos Miotto

Lavinia Feitosa

# Introdução

O problema que procuramos resolver é a falta de interação entre artistas emergentes e aqueles que gostariam de consumir seus conteúdos e a dificuldade destes artistas de se sustentar com seu trabalho.

Nosso público-alvo é qualquer pessoa que tenha algum interesse em arte, todos que gostam de apoiar trabalhadores independentes que sonham com uma vida de sucesso.

# Objetivo do aplicativo

Como o grupo do primeiro ano apontou, o Tropicalias visa auxiliar as pessoas a conseguir acesso à cultura no Brasil, especialmente à arte, criando uma comunidade unida, ativa e amigável. E auxiliar artistas de rua, artesãos, cantores e pintores independentes a viverem de suas artes, fazendo com que eles tenham reconhecimento pelo que se esforçam e gostam de fazer. O aplicativo liga essas pessoas interessadas aos artistas que gostariam de vender sua arte, assim fazendo com que os interesses de todos sejam atingidos.

# Objetivo da Análise Exploratória de Dados

O objetivo da nossa análise exploratória de dados é obter e limpar uma base de dados contendo diversas informações que poderiam influenciar o gosto de uma pessoa sobre arte e o quanto aquela pessoa gosta de arte.

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

Nós utilizaremos um modelo no nosso APP, ele será de uso administrativo e preverá se um usuário é um cliente em potencial para nosso app. O modelo receberá uma série de informações que podem influenciar no gosto de arte de um usuário, como por exemplo: renda, cidade, gênero, uso de eletrônicos etc., com isso, ele dirá se o usuário gosta muito de arte (cliente em potencial) ou se ele não gosta tanto de arte (cliente não tão bom).

# Levantamento dos dados na análise exploratória

## Busca dos dados

Para obter os dados que utilizaríamos na análise, nós optamos por criar um formulário, contendo todas as perguntas que poderiam trazer alguma informação relevante no gosto de uma pessoa sobre arte e como última pergunta o quanto essa pessoa gosta de arte.

Dessa maneira poderíamos definir se uma pessoa gosta de arte (usuário em potencial) ou se uma pessoa não tem tanto interesse em arte (usuário que não vale a pena investir).

## Justificativa de uso

Os dados que escolhemos contêm as seguintes informações sobre nossos usuários:

* Idade
* Gênero
* Renda familiar mensal
* Estado
* Cidade
* Usa muito eletrônicos (mais de 6h por dia)? (S/N)
* Grau de educação
* Pratica esportes? (S/N)
* Frequenta muito espaços públicos? (S/N)
* Gosta de arte? (S/Um pouco/N)

Todas essas informações podem influenciar tanto o grau de interesse quanto a intensidade com que uma pessoa gosta de arte. A última pergunta da pesquisa, utilizada como nossa variável de resposta, oferecia três opções aos usuários: "gostam muito de arte", "não gostam de arte" ou "gostam um pouco de arte". Entretanto, durante o processo de tratamento dos dados, agrupamos as respostas "não" na categoria "um pouco". Isso foi feito com base na suposição de que, culturalmente, as pessoas tendem a evitar afirmar que "não gostam de arte", preferindo admitir algum nível de apreciação. Além disso, algumas poderiam sentir desconforto em declarar explicitamente que não gostam de arte. Assim, ao consolidar essas duas respostas, conseguimos definir de forma mais clara o público que efetivamente demonstra um alto interesse por arte, que é o foco do estudo.

# Descrição da base de dados de trabalho

Descrição dos dados:

* Idade - Inteiro
* Gênero – *String* / Categórico:
  + - Feminino
  + - Masculino
  + - Outro
* Renda familiar mensal – *String* / Categórico:
  + - Menor ou igual à R$2400
  + - Maior que R$ 2400 e menor que R$4000
  + - Maior que R$4000 e menor que R$6500
  + - Maior que R$6500 e menor que R$8500
  + - Maior que R$8500 e menor que R$11000
  + - Maior que R$11000 e menor que R$15000
  + - Maior que R$15000
* Estado – *String* / Categórico (siglas dos estados do brasil + “outros” para outros países / SP, RJ e etc...)
* Cidade – *String*
* Usa muito eletrônicos? (mais de 6h por dia) – *Boolean*
* Grau de educação – *String* / Categórico:
  + - Nenhum
  + - Fundamental
  + - Ensino médio
  + - Ensino superior
  + - Pós-graduação
  + - Mestrado
  + - Doutorado
* Pratica esportes? (pelo menos 3 vezes por semana) - *Boolean*
* Frequenta muito espaços públicos? (parques, museus e etc) - *Boolean*
* Gosta de arte? – *String* / Categórico:
  + - Sim
  + - Um pouco
  + - Não

Gráficos Power BI (análise dados brutos):

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

# Limpeza dos dados

Mudanças realizadas:

- Exclusão da coluna “Nome”, pois seria irrelevante para a análise.

- Exclusão da coluna “Cidade”, pois chegamos à conclusão que ela “atrapalharia” a IA mais do que ajudaria.

- Valores substituídos de “um pouco” para “não” na coluna resposta (explicamos a razão anteriormente).

# Condicionamento para alimentar o modelo de ML

Fizemos o over *sampling* dos dados e para decidir qual é a melhor combinação de *encoders*, número de dimensionalidade, *hyper* parâmetros e modelos, nós automatizamos os testes utilizando *GridSearchCV* e Pipeline, após isso, analisamos os resultados e decidimos qual seria a melhor combinação.

## Condicionamento inicial

- Aplicamos o OrdinalEncoder nas colunas não numéricas.

- Aplicamos o MinMaxScaler para normalizar os dados.

- Testamos todas as dimensionalidades possíveis e chegamos a conclusão que seria melhor não reduzir nada e deixar como está.

# Definição dos objetivos e das classes

Nós utilizaremos um modelo no nosso APP, ele será de uso administrativo e preverá se um usuário é um cliente em potencial para nosso app. O modelo dirá se o usuário gosta muito de arte (cliente em potencial) ou se ele não gosta tanto de arte (cliente não tão bom).

Nossa coluna resposta é booleana e tem as classes:

1 – Gosta de arte

0 – não gosta de arte

# Definição dos modelos mais adequados para analisar os dados

Testamos 4 modelos no total:

- Naive Bayes

- Knn

- Decision Tree (com hyper parametros e com ccp alpha)

- SVC

Para avaliar a eficácia desses modelos nós os julgamos com base nas métricas de f1-score, acurácia e recall, sempre priorizando acertar a classe positiva do que a negativa (já que é o objetivo do modelo).

# Descrição dos modelos selecionados

## Modelo 1: Naive Bayes

Surpreendentemente o Bayes demonstrou resultados relativamente satisfatórios.

Utilizamos o GridSearchCV para testar a melhor combinação de encoders e dimensionalidade, medindo a performance por f1-score

Para medir a eficácia do modelo utilizamos 4 métricas principais: f1-score, acurácia, recall e a curva roc no gráfico.

## Modelo 2: KNN

O KNN demonstrou resultados não tão bons.

Utilizamos o GridSearchCV para testar a melhor combinação de encoders, dimensionalidade e nNeighbors, medindo a performance por f1-score

Para medir a eficácia do modelo utilizamos 4 métricas principais: f1-score, acurácia, recall e a curva roc no gráfico.

## Modelo 3: Decision tree por ccp alphas

A *Decision tree* por ccp alpha demonstrou resultados medianos, não foi ruim, mas, não foi bom o bastante também.

Utilizamos o *GridSearchCV* para testar a melhor combinação de *encoders,* dimensionalidade e *ccpAlphas*, medindo a performance por f1-score

Para medir a eficácia do modelo utilizamos 4 métricas principais: f1-score, acurácia, recall e a curva roc no gráfico.

## Modelo 4: Decision tree com hyperparametros

A *Decision tree* com hyper parâmetros demonstrou resultados patéticos, abaixo do esperado.

Utilizamos o *GridSearchCV* para testar a melhor combinação de *encoders,* dimensionalidade e todos os HyperParametros, medindo a performance por f1-score.

Para medir a eficácia do modelo utilizamos 4 métricas principais: f1-score, acurácia, recall e a curva roc no gráfico.

## Modelo 5: SVC

O SVC conforme esperado demonstrou resultados excepcionais, o melhor deles.

Utilizamos o *GridSearchCV* para testar a melhor combinação de *encoders,* dimensionalidade e hyperParametros, medindo a performance por f1-score

Para medir a eficácia do modelo utilizamos 4 métricas principais: f1-score, acurácia, recall e a curva roc no gráfico.

# Aplicação dos modelos selecionados

A divisão treino-teste foi feita com 20% dos dados.

Para treinar cada modelo utilizamos o GridSearchCV para testar a melhor combinação de encoders, dimensionalidade e todos os HyperParametros, medindo a performance por f1-score.

Para medir a eficácia do modelo utilizamos 4 métricas principais: f1-score, acurácia, recall e a curva roc no gráfico.

Tentamos priorizar a classe positiva do modelo, afinal, seria o mais adequado para nosso modelo, porém, ao mesmo tempo, tivemos que verificar com a curva roc se o modelo não estava "chutando” muito a classe positiva, já que, isso seria ruim.

# Análise dos Resultados

## Modelo 1: Naive Bayes

Tela preta com letras brancas

Descrição gerada automaticamente

Gráfico, Gráfico de mapa de árvore

Descrição gerada automaticamente

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

## Modelo 2: KNN

Texto

Descrição gerada automaticamente com confiança média

Gráfico, Gráfico de mapa de árvore

Descrição gerada automaticamente

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

## Modelo 3: Decision tree com CCP Alphas

Texto

Descrição gerada automaticamente

Gráfico, Gráfico de mapa de árvore

Descrição gerada automaticamente

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

## Modelo 4: Decision tree com Hyperparamentros

Uma imagem contendo Texto

Descrição gerada automaticamente

Tiveram mais hyperparametros, porém, não coube tudo no documento

Gráfico, Gráfico de mapa de árvore

Descrição gerada automaticamente

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

## Modelo 1: SVC

Texto

Descrição gerada automaticamente com confiança média

Houve mais hyperparametros testados, mas não couberam no documento.

Gráfico

Descrição gerada automaticamente

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

## Comparação Entre Modelos:

O *Naive Bayes* apresentou uma *precision* excelente para a classe 1 e uma *precision* ruim para a classe 0, porém, seu *recall* para a classe 1 foi insuficiente e seu *recall* para a classe 0 foi ok. Ao analisar a matriz de confusão podemos ver que acertou 2/3 dos registros na classe 1, o que é bom, mas, ainda está longe do ideal.

O *KNN* apresentou uma boa *precision* para a classe 1 e uma precision ruim para a classe 0, porém, seu *recall* foi mediano para a classe 1 e ok para a classe 0. Ao analisar a matriz de confusão podemos ver que ele errou muitos registros pertencentes a classe 1, o que não seria tão bom para nosso caso.

O *Decision Tree* com *CCP Alphas* apresentou uma *precision* ok para a classe 1 e uma *precision* ruim para a classe 0, porém, seu *recall* para a classe 1 foi ruim e seu *recall* para a classe 0 ok. Ao analisar a matriz de confusão podemos ver que ele “chutou” muitos registros como classe 0 e isso é confirmado pelo gráfico da curva roc.

O *Decision Tree* com Hyper parametros apresentou uma *precision* mediana para a classe 1 e uma *precision* horrível para a classe 0, além disso, seu *recall* para ambas as classes deixou muito a desejar. Ao analisar a matriz de confusão, não há um padrão obvio, ele apenas errou grande parte dos testes.

O SVC apresentou uma *precision* excelente para a classe 1 e uma *precision* ok para a classe 0, além disso, seu recall para ambas as classes foi relativamente bom. Ao analisar a matriz de confusão podemos ver que ele acertou 3/4 dos registros da classe 1 e um aproximadamente 3/4 da classe 0 também. Esses resultados se demonstraram satisfatórios e por isso, escolhemos esse modelo.

# Publicação do Modelo no APP

Percebemos que não poderíamos publicar nosso modelo de uma forma convencional, pois os dados que utilizamos no banco de dados não são os mesmos que utilizamos na IA. Afinal, não teríamos como prever se aquele usuário seria um possível cliente do nosso aplicativo a partir de seu *username*, por exemplo.

Para fins de demonstrar o funcionamento da nossa Inteligência Artificial na *Expo Tech,* criamos uma API que recebe os dados de um usuário e roda o tratamento de dados e o modelo utilizando esses dados, e retorna um *boolean* que nos diz se aquele usuário seria um cliente em potencial ou apenas um usuário comum. Para essa coleta de dados em tempo real, criamos uma página web onde os usuários preencheriam os campos com os dados necessários e automaticamente recebem a resposta sobre seu potencial como cliente ou usuário.

Porém, pensando em uma aplicação mais realista do modelo, nos obteríamos os dados dos usuários de diversas maneiras, como por exemplo comprando dados ou fazendo pesquisas, e com base nessas respostas, investiríamos pesadamente em publicidades de maneia que conseguiríamos trazer esses potenciais usuários para nosso app.

Link do GitHub da IA: <https://github.com/MocTropicalias/API_IA_INTER>

# Conclusão análise de dados

Fomos capazes de testar todas as melhorias possíveis de maneira automática e eficiente. Com base nos resultados que obtivemos, chegamos à conclusão de que o melhor modelo seria o SVC com normalização por *MinMaxScaler* e usando o PCA com o parâmetro *n\_components = None*.

Os resultados que obtivemos foram muito satisfatórios, principalmente levando em conta o tamanho de nossa base. No geral foi uma oportunidade valiosa poder aplicar diversos conceitos e técnicas de análise de dados e *machine learning* nesse projeto.

# Utilização do RPA no projeto

No projeto, temos dois bancos de dados SQL, o criado pelo segundo ano, que seria o mais completo, e o do primeiro, que foi criado exclusivamente por eles e por isso, é um pouco mais básico por conta das limitações de aprendizado. Isso foi proposto a nós como um desafio criado pelos professores.

Devido as diferenças nas duas estruturas, era necessário realizar um tratamento de dados antes de levá-los para o outro, pois algumas tabelas seriam apenas preenchidas no banco do primeiro e depois seus dados seriam levados até o do segundo, a fim de realizar isso, criamos um RPA que formata e trata todos esses dados e depois os leva de um banco ao outro.

Nós optamos por rodar esse RPA com uma carga delta, ele roda todos os dias à meia noite e passa todos os dados que foram inseridos ou modificados no dia anterior.

Código RPA: <https://drive.google.com/file/d/1Q7kV218jiiSIFMGwc9qa6cmMs-2mctdK/view?usp=sharing>

Repositório GitHub do RPA: [GitHub - MocTropicalias/RepRpaInter](https://github.com/MocTropicalias/RepRpaInter)

As primeiras 60 linhas do código são uma preparação para rodar o processo principal, onde ele importa bibliotecas, preenche variáveis, declara funções e lê arquivos. Lá, ele lê um arquivo *.env* que está apenas na AWS e obtém todas as informações sensíveis que o programa utilizará para se conectar com o banco, além disso, ele lê um arquivo JSON com informações sobre quais ambientes ele deve rodar, DEV e PRD. Ele também cria um função que serve para preencher um arquivo de log previamente criado e nomeado com a data e a hora da execução.

Após isso, o programa roda um loop que roda uma ou duas vezes – dependendo de quais ambientes ele vai rodar – onde ele trata os erros de todo o resto do código. Primeiramente, ele conecta com o banco do primeiro e do segundo no respectivo ambiente em que ele está rodando, e é onde começa o processo principal, que é dividido em quatro etapas: A primeira é passar novos registros do banco do primeiro para o do segundo, a segunda etapa passa os dados atualizados do banco do primeiro para o segundo, a terceira etapa passa registros recentemente inseridos no banco do segundo para o banco do primeiro e a quarta e última etapa passa dados recentemente atualizados do banco do segundo para o banco do primeiro.

Depois de rodar esse processo, ele encerra as conexões com os dois bancos, e se for preciso rodar novamente ele repete tudo.