Aula 10 – Engenharia de Features: Técnicas Manuais

No vasto universo da modelagem preditiva, os dados são a matéria-prima fundamental. Contudo, assim como um escultor não trabalha diretamente com um bloco de mármore bruto, um modelo de Machine Learning raramente entrega seu potencial máximo com dados em seu estado original. É aqui que entra a Engenharia de Features, a arte e a ciência de transformar dados brutos em representações que os algoritmos podem entender e, mais importante, aprender de forma eficaz.

Imagine que você está tentando prever o desempenho de um atleta. Olhar apenas para a altura e o peso pode não ser suficiente. Mas e se você criasse uma nova "feature" como o Índice de Massa Corporal (IMC), ou a "velocidade média nos últimos 100 metros"? Essas novas informações, derivadas dos dados originais, são muito mais preditivas e reveladoras. A Engenharia de Features é exatamente isso: a criação de variáveis mais informativas para o seu modelo.

Nesta aula, nosso objetivo é mergulhar nas técnicas manuais de Engenharia de Features, capacitando você a extrair o máximo valor dos seus dados. Ao final, você será capaz de identificar oportunidades de criação de features a partir de datas, gerar termos polinomiais e de interação, aplicar transformações não lineares e agrupar variáveis numéricas, tudo para construir modelos mais robustos e precisos. Prepare-se para refinar sua matéria-prima e elevar a inteligência dos seus modelos.

O Tempo é Ouro: Extraindo Valor de Datas e Horas

> **Por que transformar dados temporais?**
>
> Para um algoritmo de ML, uma data como "2023-10-26" é apenas uma sequência de caracteres sem significado preditivo direto.

Frequentemente, nossos conjuntos de dados contêm informações temporais, como datas de transações, horários de eventos ou períodos de registro. No entanto, para um algoritmo de Machine Learning, uma data como "2023-10-26" ou um horário como "14:35:00" são apenas sequências de caracteres ou números que, por si só, não carregam um significado preditivo direto. O desafio é transformar essa informação bruta em algo que o modelo possa realmente aprender e utilizar.

Pense em como nós, humanos, interpretamos o tempo. Não vemos apenas uma sequência de números, mas sim padrões: "é fim de semana?", "é horário de pico?", "é um feriado?". Cada um desses aspectos temporais influencia nosso comportamento e, consequentemente, os fenômenos que tentamos modelar. A Engenharia de Features nos permite "ensinar" esses padrões aos nossos modelos, desvendando o poder oculto por trás de cada carimbo de data/hora.

Ao extrair componentes específicos de datas e horas, podemos criar novas variáveis que capturam a sazonalidade, tendências e ciclos que são cruciais para a previsão. Por exemplo, saber se uma transação ocorreu em um dia útil ou em um fim de semana pode ser muito mais relevante para prever o volume de vendas do que a data exata. Essa abordagem nos permite ir além do óbvio e construir uma representação mais rica do tempo.

Desdobrando o Calendário: Features Temporais Essenciais

Uma das formas mais diretas de extrair valor de datas e horas é desmembrá-las em seus componentes constituintes. Cada parte de uma data ou hora pode se tornar uma feature independente, revelando padrões que seriam invisíveis de outra forma. Por exemplo, o ano pode indicar uma tendência de longo prazo, o mês pode capturar sazonalidade anual, e o dia da semana pode revelar padrões semanais.

Componentes Básicos

- Ano
- Mês
- Dia
- Hora
- Minuto
- Dia da semana

Features Derivadas

- Dia do ano
- Semana do ano
- Trimestre
- É feriado?
- É fim de semana?

Diferenças Temporais

- Tempo desde última compra
- Idade do cliente
- Dias até evento
- Duração da sessão

Imagine que você está analisando dados de tráfego em uma cidade. O número de carros na rua não é o mesmo em uma segunda-feira de manhã e em um domingo à tarde. Ao criar features como dia_da_semana, hora_do_dia e mes_do_ano, você está fornecendo ao modelo informações contextuais valiosas. Essas features permitem que o algoritmo identifique que, por exemplo, as terças-feiras às 8h da manhã têm um padrão de tráfego diferente das quintas-feiras às 20h.

Além dos componentes básicos, podemos criar features mais complexas, como dia_do_ano, semana_do_ano, trimestre ou até mesmo eh_feriado (uma variável binária). A diferença entre duas datas também é uma feature poderosa, como o tempo_desde_ultima_compra ou a idade_do_cliente. Essas variáveis transformam uma simples data em um conjunto de informações ricas e preditivas, essenciais para modelos que lidam com séries temporais ou eventos sequenciais.

Capturando Ciclos e Eventos Especiais

Natureza Cíclica do Tempo

A representação de features temporais vai além da simples extração de componentes. Alguns aspectos do tempo são cíclicos, como o dia da semana ou o mês do ano. Se representarmos o dia da semana como números (1 para segunda, 7 para domingo), o modelo pode inferir uma relação ordinal que não existe (domingo não é "7 vezes mais" que segunda, e a distância entre domingo e segunda é a mesma que entre segunda e terça). Para capturar a natureza cíclica, transformações senoidais e cossenoidais são frequentemente utilizadas, mapeando os valores para um círculo.

Eventos Especiais

Considere o impacto de feriados e eventos especiais. Um feriado nacional ou um grande evento esportivo pode alterar drasticamente o comportamento que estamos tentando prever, como o consumo de energia ou o fluxo de clientes em uma loja. Criar uma feature binária (eh_feriado) ou categórica (tipo_de_feriado) permite que o modelo aprenda o efeito específico desses dias. Essa é uma forma de injetar conhecimento de domínio diretamente nos dados.

> **Insight Prático:** Essas técnicas são particularmente úteis em cenários como a previsão de demanda, onde a sazonalidade e os eventos especiais são fatores determinantes. Ao fornecer ao modelo uma representação precisa desses ciclos e interrupções, aumentamos significativamente sua capacidade de fazer previsões acuradas. É como dar ao modelo um calendário anotado com todos os eventos importantes, em vez de apenas uma lista de datas.

Elevando o Jogo: Features Polinomiais para Relações Não Lineares

O Problema da Linearidade

Muitas vezes, a relação entre uma feature e a variável alvo não é linear. Um modelo linear, por sua natureza, tenta ajustar uma linha reta aos dados.

Muitas vezes, a relação entre uma feature e a variável alvo não é linear. Um modelo linear, por sua natureza, tenta ajustar uma linha reta aos dados. No entanto, na realidade, os fenômenos podem seguir curvas, parábolas ou outras formas mais complexas. Se ignorarmos essa não linearidade, nosso modelo pode subestimar ou superestimar o impacto de uma variável em diferentes faixas de valores, resultando em previsões imprecisas.

01

Feature Original

Começamos com uma feature X (ex: tempo)

02

Adicionar Potências

Criamos X^2, X^3, etc.

03

Modelo Aprende Curvas

O modelo linear combina essas potências para formar funções não lineares

A solução para esse problema, dentro do escopo da engenharia manual, é a criação de features polinomiais. Basicamente, isso envolve adicionar potências de features existentes ao nosso conjunto de dados. Se temos uma feature X, podemos criar X^2 (X ao quadrado), X^3 (X ao cubo) e assim por diante. Ao incluir essas novas features, estamos permitindo que um modelo linear capture relações curvas, pois ele pode agora combinar essas potências para formar uma função não linear.

Imagine que você está modelando o crescimento de uma planta ao longo do tempo. No início, o crescimento pode ser lento, depois acelerar e, eventualmente, desacelerar. Uma linha reta nunca capturaria esse padrão. Mas se você adicionar tempo^2 e tempo^3 como features, o modelo linear pode "dobrar" sua linha para se ajustar à curva de crescimento, como se estivesse usando uma régua flexível para traçar o caminho. Essa técnica é uma ponte poderosa entre a simplicidade dos modelos lineares e a complexidade das relações do mundo real.

A Sinergia dos Dados: Features de Interação

O que são Features de Interação?

Features de interação são criadas multiplicando-se duas ou mais features existentes. Se temos as features X e Y, podemos criar uma nova feature `X * Y`.

Essa nova feature representa a "sinergia" ou "antagonismo" entre X e Y.

Por que são importantes?

Além das relações não lineares de uma única variável, é comum que o efeito de uma feature sobre a variável alvo dependa do valor de outra feature. Por exemplo, um desconto pode ter um impacto nas vendas que varia significativamente se o produto está em alta demanda ou em baixa demanda. Modelos que não consideram essas interações podem perder nuances importantes e fazer previsões menos precisas.

Exemplo: Culinária

O sabor de um prato não é apenas a soma dos sabores de seus ingredientes; a forma como eles interagem e se combinam cria um perfil de sabor totalmente novo.

Exemplo: Clima

A interação entre temperatura e umidade pode ter um efeito diferente na previsão de chuva do que cada uma dessas variáveis isoladamente.

Exemplo: E-commerce

O efeito de um desconto nas vendas pode depender da categoria do produto e da época do ano.

Pense em uma receita culinária. O sabor de um prato não é apenas a soma dos sabores de seus ingredientes; a forma como eles interagem e se combinam cria um perfil de sabor totalmente novo. Da mesma forma, a interação entre temperatura e umidade pode ter um efeito diferente na previsão de chuva do que cada uma dessas variáveis isoladamente. Ao criar features de interação, estamos dando ao nosso modelo a capacidade de "sentir" essas combinações e entender como as variáveis se influenciam mutuamente, revelando padrões mais profundos nos dados.

Domando a Assimetria: Transformações Não Lineares

O Desafio das Distribuições Assimétricas

Variáveis como renda, número de acessos a um site ou tempo de espera frequentemente exibem assimetria e são propensas a outliers.

Muitos algoritmos de Machine Learning, especialmente os modelos lineares e baseados em distância, assumem que as features seguem uma distribuição normal ou são simetricamente distribuídas. No entanto, na prática, é muito comum encontrar variáveis com distribuições assimétricas, onde os dados se acumulam em um lado e têm uma "cauda" longa para o outro. Variáveis como renda, número de acessos a um site ou tempo de espera frequentemente exibem essa assimetria, além de serem propensas a outliers.

Quando uma feature é muito enviesada, ela pode dominar o modelo, ou os outliers podem distorcer os coeficientes e as relações aprendidas. As transformações não lineares são uma ferramenta poderosa para mitigar esses problemas. Elas aplicam uma função matemática à feature para alterar sua distribuição, tornando-a mais simétrica e reduzindo o impacto de valores extremos. O objetivo é "comprimir" a cauda longa da distribuição, trazendo os valores mais distantes para mais perto da média.

1

Distribuição Original

Dados enviesados com cauda longa

2

Aplicar Transformação

Log, raiz quadrada, etc.

3

Distribuição Normalizada

Dados mais simétricos e estáveis

Imagine que você tem uma lente especial que pode "achatar" uma paisagem montanhosa muito íngreme, tornando-a mais suave e fácil de percorrer. As transformações não lineares fazem algo semelhante com a distribuição dos seus dados. Ao aplicar funções como o logaritmo ou a raiz quadrada, podemos estabilizar a variância, reduzir a assimetria e, em muitos casos, melhorar o desempenho do modelo, tornando os dados mais "palatáveis" para os algoritmos.

Logaritmos e Raízes: Ferramentas para Normalizar Dados

Transformação Logarítmica

A transformação logarítmica é particularmente útil para variáveis que são estritamente positivas e exibem um forte viés à direita, ou seja, a maioria dos valores é pequena, mas há alguns valores muito grandes. Ao aplicar o logaritmo, a escala da variável é drasticamente reduzida, e a distribuição tende a se tornar mais simétrica.

Exemplo: Se você tem dados de salários onde a maioria das pessoas ganha um valor médio, mas algumas poucas têm salários exorbitantes, a distribuição será enviesada. Aplicar `log(salário)` pode transformar essa distribuição em algo mais próximo de uma curva normal.

Importante: Para o logaritmo, os valores devem ser positivos. Se houver zeros ou negativos, é comum adicionar uma pequena constante (ex: `log(X+1)`).

Transformação de Raiz Quadrada

A transformação de raiz quadrada é uma alternativa menos agressiva que o logaritmo, também aplicável a variáveis positivas. Ela é útil quando o viés não é tão extremo, mas ainda é necessário reduzir a assimetria e a influência de outliers.

Trade-off: Ambas as transformações, embora poderosas, alteram a interpretabilidade direta da feature, pois agora estamos trabalhando com a "raiz quadrada da idade" ou o "logaritmo do salário". No entanto, o ganho em desempenho do modelo frequentemente compensa essa pequena perda de interpretabilidade direta da feature transformada.

> **Dica Prática:** Entre as transformações não lineares mais comuns, o logaritmo (log) e a raiz quadrada se destacam pela sua simplicidade e eficácia.

Agrupando para Clareza: O Poder do Binning (Agrupamento)

Variáveis numéricas contínuas, especialmente aquelas com uma grande variedade de valores únicos ou muitos outliers, podem apresentar desafios para alguns modelos de Machine Learning. A granularidade excessiva pode introduzir ruído, e a presença de valores extremos pode distorcer as relações. Nesses casos, uma técnica eficaz de engenharia de features é o agrupamento, ou *binning*, que consiste em transformar uma variável numérica contínua em uma variável categórica (ordinal) dividindo-a em faixas ou "bins".

Exemplo: Idade

Em vez de usar a idade exata (18-90 anos), agrupe em faixas:

- **Jovens:** 18-25
- **Adultos:** 26-45
- **Maduros:** 46-65
- **Idosos:** 66+

Benefícios do Binning

- Foco em padrões gerais
- Redução de sensibilidade a variações
- Melhor tratamento de outliers
- Linearização de relações não lineares
- Redução de ruído

Imagine que você está analisando a idade dos clientes. Em vez de usar a idade exata (que pode variar de 18 a 90 anos, por exemplo), você pode agrupá-la em faixas etárias: "jovens" (18-25), "adultos" (26-45), "maduros" (46-65) e "idosos" (66+). Essa simplificação pode ajudar o modelo a focar em padrões mais gerais, reduzir a sensibilidade a pequenas variações e até mesmo lidar melhor com outliers, que agora caem em uma categoria em vez de serem um ponto isolado.

O binning pode ser uma ferramenta poderosa para linearizar relações não lineares, reduzir o ruído nos dados e até mesmo para lidar com valores ausentes, atribuindo-os a uma categoria específica. É como organizar uma biblioteca: em vez de procurar um livro pelo título exato em uma prateleira desorganizada, você o encontra mais facilmente se os livros estiverem agrupados por gênero ou autor. Essa organização simplifica a busca e a compreensão dos padrões.

Estratégias de Binning: Escolhendo o Melhor Agrupamento

Existem diversas abordagens para realizar o binning, cada uma com suas vantagens e desvantagens. A escolha do método depende das características dos seus dados e dos objetivos do seu modelo.

1

Binning de Largura Fixa (Equal-width)

Divide o intervalo da variável em bins de tamanhos iguais. Por exemplo, se a idade varia de 0 a 100, e você quer 10 bins, cada bin terá 10 anos de largura (0-10, 11-20, etc.). É simples de implementar, mas pode resultar em bins vazios ou com poucos dados se a distribuição for muito enviesada.

2

Binning de Frequência Fixa (Equal-frequency / Quantiles)

Divide a variável de forma que cada bin contenha aproximadamente o mesmo número de observações. Por exemplo, se você quer 4 bins (quartis), cada um terá 25% dos dados. Isso garante que todos os bins tenham representatividade, mas os intervalos de cada bin podem variar bastante em largura.

3

Binning Baseado em Modelos (ex: Decision Tree)

Utiliza um modelo (como uma árvore de decisão) para encontrar os pontos de corte que melhor separam a variável alvo. Este método é mais sofisticado e pode criar bins que são mais preditivos, mas é mais complexo e pode levar a overfitting se não for bem controlado.

Conceito	Âmbito/Aplicação	Base/Origem	Exemplo
Equal-width Binning	Distribuições uniformes ou pouco enviesadas	Divisão do range em intervalos iguais	Idade: 0-20, 21-40, 41-60, 61-80, 81-100
Equal-frequency Binning	Distribuições enviesadas ou com outliers	Divisão dos dados em grupos com mesma contagem	Renda: 25% menores, 25% médios-baixos, etc.

Decisão Crítica

A decisão sobre quantos bins criar e qual método usar é crucial. Poucos bins podem levar à perda de informações importantes, enquanto muitos bins podem reintroduzir o ruído que você estava tentando eliminar. A validação cruzada e a análise do impacto no desempenho do modelo são essenciais para otimizar essa escolha.

Engenharia Manual e a Interpretabilidade (XAI): Uma Aliança Necessária

O Desafio da Caixa Preta

Na era da Inteligência Artificial Explicável (XAI), a capacidade de entender e justificar as decisões de um modelo de Machine Learning tornou-se tão importante quanto sua precisão. Modelos complexos, como redes neurais profundas ou ensembles de árvores, são frequentemente vistos como "caixas pretas", dificultando a compreensão de como chegam às suas previsões. É aqui que a engenharia de features manual desempenha um papel surpreendentemente relevante.

Features e Simplicidade

Features bem engenheiradas podem, paradoxalmente, tornar modelos mais simples e, portanto, mais interpretáveis. Se você consegue criar uma feature que encapsula uma relação complexa de forma direta (por exemplo, um IMC em vez de peso e altura separados), um modelo linear pode ser capaz de aprender essa relação, que de outra forma exigiria um modelo muito mais complexo. Isso significa que a engenharia manual pode, em alguns casos, reduzir a necessidade de modelos intrinsecamente opacos.

Features Significativas

Conceitos compreensíveis como dia_da_semana ou log_salario

Técnicas XAI

SHAP e LIME funcionam melhor com features bem construídas

Explicações Acionáveis

Resultados mais compreensíveis para stakeholders

Mesmo quando usamos modelos complexos, a qualidade das features é fundamental para a XAI. Técnicas como SHAP (SHapley Additive exPlanations) e LIME (Local Interpretable Model-agnostic Explanations) nos ajudam a entender a contribuição de cada feature para uma previsão específica. Se as features foram cuidadosamente construídas e representam conceitos significativos (como dia_da_semana ou log_salario), as explicações geradas pela XAI serão muito mais compreensíveis e acionáveis. A engenharia manual, portanto, não é apenas sobre precisão, mas também sobre construir uma base sólida para a interpretabilidade.

A Engenharia Manual na Era do AutoML: Ainda Relevante?

A Pergunta do Momento

Com o avanço das plataformas de AutoML, que prometem automatizar grande parte do ciclo de vida do ML, a engenharia de features manual ainda é relevante?

Com o avanço das plataformas de Automação de Machine Learning (AutoML), que prometem automatizar grande parte do ciclo de vida do ML, desde o pré-processamento até a seleção e otimização de modelos, surge a pergunta: a engenharia de features manual ainda é relevante? A resposta é um retumbante sim. Embora o AutoML possa automatizar a criação de algumas features genéricas (como interações polinomiais básicas ou transformações logarítmicas), ele raramente consegue replicar a inteligência e o conhecimento de domínio de um especialista humano.

AutoML: O Chef Robótico

Pode seguir milhares de receitas e otimizar ingredientes para pratos existentes. É eficiente e produz resultados consistentes.

Engenharia Manual: O Chef Experiente

Cria novas receitas inovadoras, adapta pratos para restrições específicas e inventa combinações que ninguém havia pensado antes.

Pense no AutoML como um chef robótico que pode seguir milhares de receitas e otimizar ingredientes para pratos existentes. Ele é eficiente e pode produzir resultados consistentes. No entanto, um chef humano experiente, com profundo conhecimento dos ingredientes, culturas culinárias e paladares, é quem cria novas receitas inovadoras, adapta pratos para restrições dietéticas específicas ou inventa combinações de sabores que ninguém havia pensado antes.

> **Exemplos de Features de Domínio:**
>
> - `tempo_desde_ultima_interacao` em um sistema de CRM
> - `proporcao_de_palavras_negativas` em análise de sentimento
> - `taxa_de_conversao_por_categoria` em e-commerce

Da mesma forma, a engenharia de features manual permite a criação de features altamente específicas e contextuais, que dependem de um profundo entendimento do problema de negócio e dos dados. Features como tempo_desde_ultima_interacao em um sistema de CRM, ou proporcao_de_palavras_negativas em análise de sentimento, são exemplos de inteligência de domínio que o AutoML dificilmente descobriria por conta própria. A engenharia manual complementa o AutoML, permitindo que os cientistas de dados foquem em insights de alto nível, enquanto as ferramentas automatizadas cuidam das tarefas mais rotineiras.

Boas Práticas e Armadilhas na Engenharia de Features

A engenharia de features é uma ferramenta poderosa, mas como qualquer ferramenta, seu uso exige sabedoria e atenção para evitar armadilhas comuns que podem comprometer a qualidade do seu modelo.

Entendimento do Domínio

As melhores features geralmente nascem de insights sobre o negócio ou o fenômeno que está sendo modelado. Conversar com especialistas da área, analisar dados exploratoriamente e formular hipóteses são passos cruciais antes de começar a criar features.

Validação Cruzada

Sempre avalie o impacto das suas novas features usando técnicas de validação robustas. Uma feature que parece boa no conjunto de treino pode não generalizar bem para dados não vistos.

Evitar Data Leakage

É vital evitar o vazamento de dados, onde informações do conjunto de teste ou do futuro são inadvertidamente usadas para criar features no conjunto de treino. Por exemplo, calcular a média de uma variável usando dados que incluem o próprio ponto que você está tentando prever.

Armadilhas Comuns

Overfitting

A criação excessiva de features complexas pode fazer com que o modelo memorize o conjunto de treino em vez de aprender padrões generalizáveis.

Features Redundantes

Criar features redundantes pode aumentar a complexidade do modelo sem adicionar valor preditivo.

Custo Computacional

Features muito complexas ou em grande número podem tornar o treinamento e a inferência do modelo lentos e caros.

> A chave é encontrar o equilíbrio entre a riqueza das features e a simplicidade do modelo.

Ferramentas Essenciais para a Engenharia de Features

A implementação prática das técnicas de engenharia de features manuais é facilitada por bibliotecas e ferramentas amplamente utilizadas no ecossistema de ciência de dados. Dominar essas ferramentas é crucial para transformar conceitos em código funcional.

Pandas

A biblioteca **Pandas** é a espinha dorsal para a manipulação de dados em Python. Ela oferece funcionalidades robustas para trabalhar com DataFrames, permitindo a criação de novas colunas a partir de operações em colunas existentes.

Funcionalidades principais:

- Métodos `.dt` para componentes de data/hora
- Operações aritméticas entre colunas
- Criação de features polinomiais e de interação
- Transformações customizadas com `.apply()`

Scikit-learn

Para transformações mais complexas e padronizadas, a biblioteca **Scikit-learn** é indispensável. Ela oferece módulos específicos para engenharia de features:

- `PolynomialFeatures`: Gera features polinomiais e de interação automaticamente
- `PowerTransformer`: Aplica transformações Box-Cox e Yeo-Johnson
- `KBinsDiscretizer`: Realiza binning com várias estratégias
- `FunctionTransformer`: Aplica funções customizadas

Dados Brutos

Começar com dados em formato tabular

Pandas + Scikit-learn

Aplicar transformações e criar features

Features Otimizadas

Dados prontos para modelagem preditiva

A combinação dessas ferramentas permite que o cientista de dados implemente de forma eficiente e escalável as técnicas de engenharia de features discutidas, transformando dados brutos em um formato otimizado para a modelagem preditiva.

Consolidação e Próximos Passos

> **Recapitulando a Jornada**
>
> Chegamos ao final de uma jornada crucial na modelagem preditiva: a Engenharia de Features Manual.

Chegamos ao final de uma jornada crucial na modelagem preditiva: a Engenharia de Features Manual. Vimos que transformar dados brutos em representações mais informativas não é apenas uma etapa de pré-processamento, mas uma arte que pode desbloquear o verdadeiro potencial dos seus modelos. Desde a extração de padrões temporais até a criação de relações não lineares e a organização de dados contínuos em categorias, cada técnica visa aprimorar a capacidade do algoritmo de aprender e generalizar. A engenharia de features é o elo que conecta o conhecimento de domínio com o poder computacional, permitindo que construamos modelos mais precisos, robustos e, cada vez mais, interpretáveis.

Em Prática

Comece sempre com uma análise exploratória profunda dos seus dados. Identifique variáveis temporais e pense em como o tempo influencia seu problema. Observe distribuições enviesadas e considere transformações. Busque interações lógicas entre variáveis que podem ser capturadas por features de interação. E, acima de tudo, valide suas novas features com cuidado para garantir que elas realmente adicionam valor ao seu modelo sem introduzir ruído ou vazamento de dados.

Autoavaliação

1. **Qual das seguintes técnicas é mais adequada para transformar uma variável numérica contínua com uma distribuição fortemente enviesada à direita (ex: renda) em uma forma mais simétrica para um modelo linear?**
   - a) Criação de features polinomiais
   - b) Agrupamento (Binning) de largura fixa
   - c) Transformação logarítmica
   - d) Criação de features de interação
2. **Ao criar features a partir de datas, qual a principal razão para usar transformações senoidais/cossenoidais para o dia da semana ou mês do ano?**
   - a) Para reduzir o número de categorias e simplificar o modelo.
   - b) Para capturar a natureza cíclica da variável, evitando uma interpretação ordinal.
   - c) Para lidar com valores ausentes em colunas de data.
   - d) Para aumentar a interpretabilidade da feature para o usuário final.
3. **Um cientista de dados está construindo um modelo para prever o preço de imóveis. Ele percebe que o efeito do "número de quartos" no preço é diferente dependendo se o imóvel está localizado em uma "área nobre" ou não. Qual técnica de engenharia de features seria mais apropriada para capturar essa relação?**
   - a) Transformação de raiz quadrada no número de quartos.
   - b) Agrupamento (Binning) do número de quartos.
   - c) Criação de uma feature de interação entre "número de quartos" e "área nobre".
   - d) Criação de features polinomiais para o número de quartos.
4. **Qual das seguintes afirmações melhor descreve a relação entre Engenharia de Features manual e AutoML?**
   - a) AutoML substitui completamente a necessidade de engenharia de features manual.
   - b) Engenharia de features manual é obsoleta na era do AutoML.
   - c) AutoML pode automatizar algumas técnicas, mas a engenharia manual é crucial para features de domínio específico e insights contextuais.
   - d) Ambas as abordagens são mutuamente exclusivas e não devem ser usadas em conjunto.
5. **Explique como a engenharia de features manual pode contribuir para a interpretabilidade de modelos de Machine Learning, mesmo na presença de técnicas de XAI.**

Gabarito:

1. c)
2. b)
3. c)
4. c)

Próxima Aula

Aula 11 – Seleção de Features: Métodos de Filtro e Wrapper

Exploraremos como escolher as features mais relevantes para o seu modelo, otimizando o desempenho e a interpretabilidade.

Recursos Adicionais:

- **Livro:** "Feature Engineering for Machine Learning" por Alice Zheng e Amanda Casari – Para aprofundar nas técnicas e conceitos.
- **Artigo:** "A Gentle Introduction to Feature Engineering" (Kaggle Learn) – Para uma revisão prática e exemplos de código.
- **Curso Online:** "Feature Engineering for Machine Learning" (Coursera/Udemy) – Para prática hands-on com diferentes datasets.