**Relatório 10 - Prática: Lidando com Dados do Mundo Real (II)**

Ronny Gabryel Colatino de Souza

**Descrição da atividade**

# K-Nearest Neighbors (KNN)

Nesse card começou introduzindo o algoritmo K-Nearest Neighbors ou KNN que é usado pra classificar novos pontos de dados com base na distância de dados conhecidos. Realmente ele é um dos modelos mais simples que existe mas mesmo assim ainda se encaixa como aprendizado supervisionado achei interessante porque não precisa de treinamento complexo ele só calcula distâncias e pronto

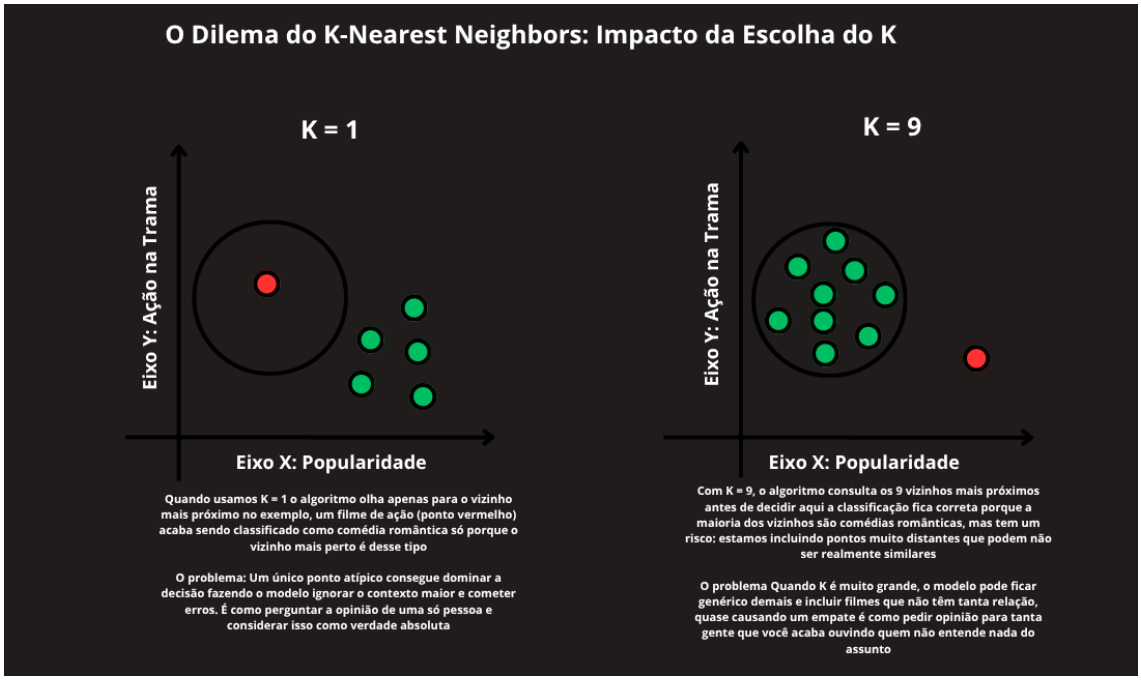
A ideia do KNN é bem direta imagina que você tem vários pontos de dados já classificados tipo usuários que gostam de determinados filmes ou produtos quando chega um novo ponto o algoritmo olha pros K vizinhos mais próximos dele e classifica baseado no que a maioria desses vizinhos são o "K" é justamente o número de vizinhos que você vai considerar, tipo se K=3, ele olha pros 3 pontos mais próximos, se K=5, olha pros 5 mais próximos e assim vai

O instrutor mostrou um exemplo bem legal criando um sistema de similaridade de filmes baseado apenas em metadados tipo gênero e popularidade primeiro foi carregado o dataset MovieLens com todas as avaliações de filmes num DataFrame do Pandas usando as bibliotecas pandas e numpy a tabela tinha três colunas principais user_id, movie_id e rating basicamente mostrando qual usuário avaliou qual filme e que nota deu

Depois os dados foram agrupados pelo ID do filme usando groupby pra calcular duas coisas importantes quantas avaliações cada filme teve que seria tipo a popularidade dele e a média das notas que ele recebeu com isso dá pra recomendar filmes similares a outros que o usuário já gostou tipo se você curtiu Star Wars o algoritmo vai achar outros filmes de ficção científica com popularidade parecida

Uma coisa que ficou clara é que a escolha da métrica de distância é super importante no caso foi usada distância cosseno pros gêneros e diferença absoluta pra popularidade mas dá pra usar outras métricas tipo distância euclidiana, Manhattan, etc dependendo do

problema também o valor de K faz diferença se você escolhe um K muito pequeno tipo 1 ou 2 o resultado pode ser muito influenciado por outliers mas se escolhe um K muito grande pode acabar pegando filmes que não são tão similares assim




O exemplo dos filmes mostrou bem como metadados podem ser usados pra criar sistemas de recomendação básicos claro que os sistemas reais tipo Netflix, Amazon e Spotify são bem mais complexos e usam muito mais informações como histórico de visualização padrões de comportamento, horários, essas coisas, mas o conceito base é parecido com o que foi mostrado aqui

## Principal Component Analysis (PCA)

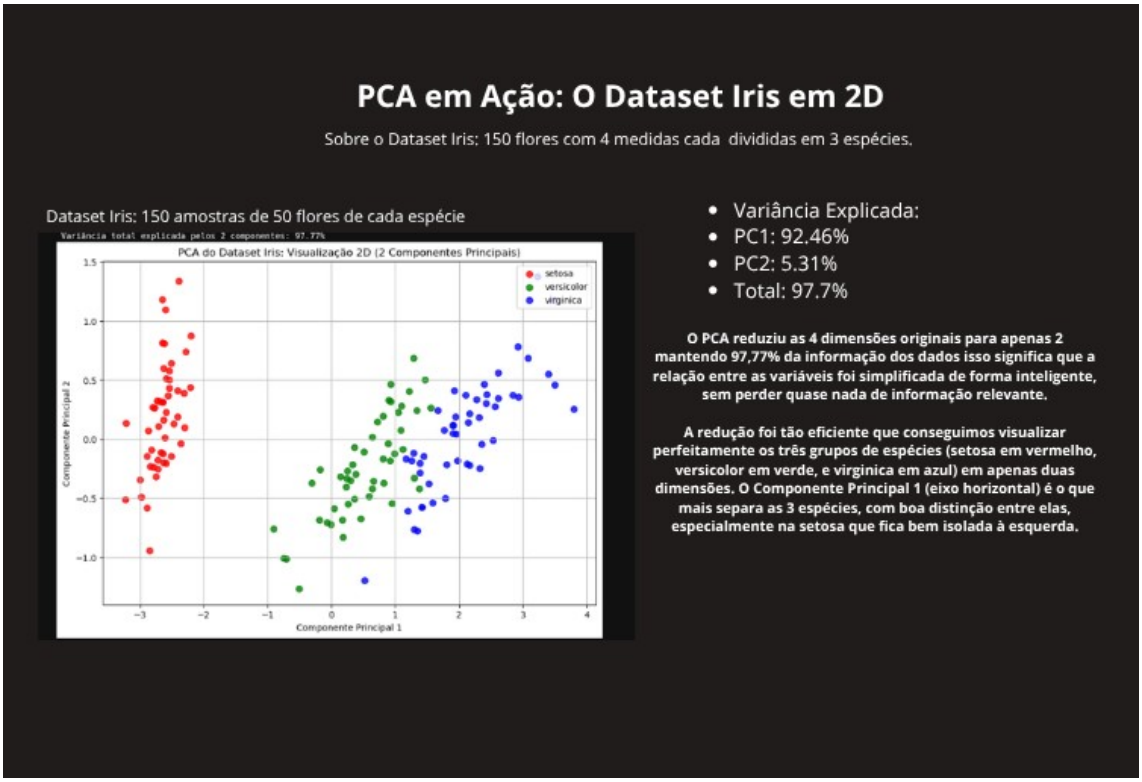
O PCA é uma técnica de redução de dimensionalidade basicamente ela permite pegar dados com várias dimensões e reduzir pra menos dimensões mantendo o máximo possível da variância dos dados originais achei esse conceito meio doido no começo

porque tipo como você joga fora informação mas mantém os dados úteis mas depois fez bastante sentido

A ideia é que muitas vezes os dados têm informações redundantes ou correlacionadas por exemplo se você tá medindo a altura e o peso de pessoas provavelmente essas duas coisas estão relacionadas então não precisa necessariamente das duas dimensões pra entender o padrão geral O PCA encontra as direções onde os dados variam mais os componentes principais e projeta tudo nessas direções.

O exemplo usado foi o dataset Iris que vem junto com o scikit-learn é uma coleção pequena de dados que tem quatro dimensões pra três tipos diferentes de flores Iris o comprimento e largura das pétalas e sépalas de várias flores individuais de cada espécie. O dataset tem 150 amostras no total, 4 features dimensões e 3 espécies diferentes setosa, versicolor e virginica

O que o PCA faz é pegar esse conjunto de dados 4D e projetar ele em vetores ortogonais que formam a base da nova projeção 2D embora não dê pra visualizar muito bem o que esses vetores 4D significam o importante é o resultado mesmo jogando fora duas das quatro dimensões o PCA conseguiu preservar 92% da variância dos dados em apenas uma dimensão e o segundo componente adiciona mais 5% totalizando 97,77% da variância preservada




Isso é bem impressionante porque significa que perdemos menos de 3% da informação ao reduzir de 4 dimensões pra 2 provavelmente funciona tão bem porque o tamanho geral de uma flor faz com que tanto as pétalas quanto as sépalas aumentem proporcionalmente então o PCA conseguiu capturar essa relação de proporção entre largura e altura sozinho

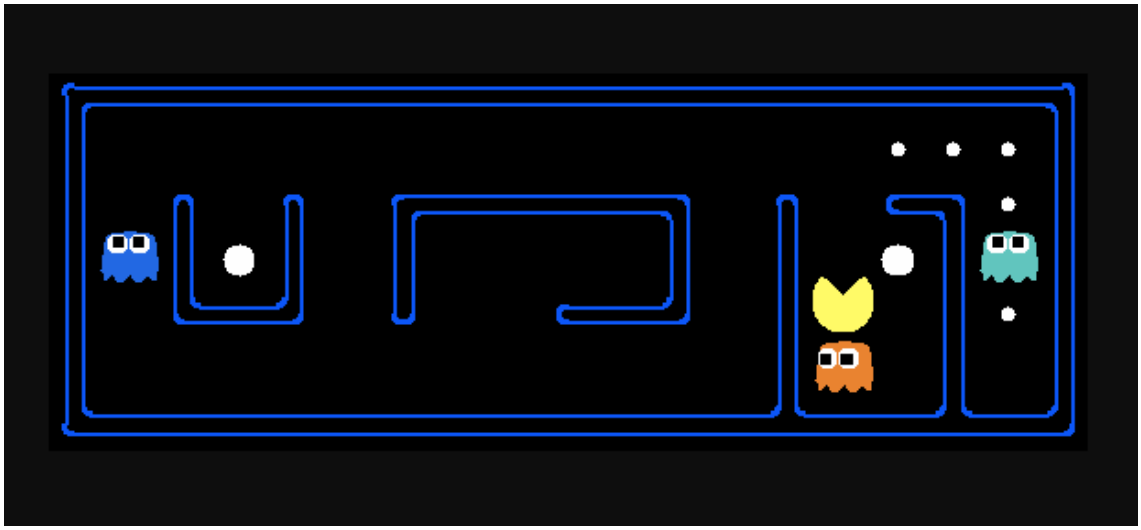
Na prática o PCA é muito usado em reconhecimento facial compressão de imagens análise de dados genômicos e basicamente qualquer lugar onde você tem muitas features e precisa simplificar sem perder informação importante tipo uma imagem em preto e branco tem três dimensões de dados posição X, posição Y e brilho em cada ponto e dá pra usar PCA pra comprimir mantendo a essência da imagem

## Q-Learning

O Q-Learning é um tipo de aprendizado por reforço que é bem diferente do aprendizado supervisionado que a gente viu antes a ideia geral é que você tem um agente que precisa explorar um ambiente e aprender a tomar as melhores decisões baseado em recompensas e punições

O "Q" representa o valor de qualidade de uma ação em um determinado estado basicamente o algoritmo vai construindo uma tabela chamada Q-table que diz se eu tô no estado X e faço a ação Y qual é o valor esperado que vou receber conforme o agente vai explorando o ambiente ele vai atualizando esses valores baseado no que acontece.

Um exemplo clássico que achei bem legal é o do Pac-Man tá numa posição específica do labirinto esse é o estado ele pode tomar algumas ações ir pra cima, pra baixo, esquerda, direita. Cada uma dessas ações tem um valor Q associado. Se ir pra cima leva ele perto de um fantasma o Q vai ser negativo tipo -10 se ir pra direita leva ele perto de uma bolinha de pontos o Q vai ser positivo tipo +5



Uma coisa interessante é que se o algoritmo sempre escolher o Q mais alto ele pode acabar perdendo caminhos potencialmente melhores que ele ainda não explorou direito por isso é comum adicionar uma variável de exploração geralmente chamada de epsilon ou μ se um número aleatório for menor que esse valor o algoritmo escolhe uma ação aleatória ao invés da melhor conhecida isso garante que ele continue explorando novas possibilidades

## K-Fold Cross-Validation

Esse é um método pra validar modelos de machine learning e evitar overfitting que é quando o modelo decora os dados de treino mas não consegue generalizar pra dados novos achei esse conceito super importante porque de que adianta ter um modelo que acerta 100% no treino mas erra tudo na vida real

A ideia do K-Fold é dividir seus dados em K partes no caso folds de tamanho aproximadamente igual aí você treina o modelo K vezes e cada vez usa uma parte diferente como teste e o resto como treino

No final, você tem K resultados diferentes e aí tira a média deles pra ter uma ideia mais confiável de como o modelo realmente performa isso é muito melhor do que simplesmente dividir em treino barr a teste uma vez só porque pode ser que você tenha dado sorte ou azar na divisão.

O valor de K precisa ser escolhido com cuidado um K muito pequeno tipo 2 ou 3 significa que você vai treinar com poucos dados, o que pode não representar bem o dataset todo um K muito grande tipo 20 ou 30 significa que cada rodada de treino vai ser muito parecida com as outras então você não ganha tanta variabilidade o mais comum é usar K=5 ou K=10 que geralmente dá um bom equilíbrio

Na prática quando aplicamos K-Fold no algoritmo KNN por exemplo dá pra perceber como a escolha do K do KNN não do K-Fold afeta o resultado com K-Fold você consegue testar vários valores de K e ver qual realmente funciona melhor pro seu problema específico não só pros seus dados de treino.

Existem variações do K-Fold também tipo o Stratified K-Fold que mantém a proporção de classes em cada fold útil e o Leave-One-Out que é tipo um K-Fold onde K é igual ao número total de exemplos

## Outliers

Outliers são aqueles pontos de dados que fogem completamente do padrão tipo o salário de um bilionário no meio de dados de cidadãos comuns ou uma pessoa de 2,30m de altura num dataset de altura de adultos detectar e lidar com outliers é crucial porque eles podem bagunçar completamente suas análises e modelos

O interessante é que nem todo outlier é ruim ou erro as vezes eles representam eventos raros mas reais que são justamente o que você quer estudar tipo detectar fraudes em transações bancárias as fraudes são outliers mas são exatamente o que você quer identificar então antes de simplesmente remover outliers é importante entender de onde eles vêm

Existem várias técnicas pra detectar outliers tipo

**Visualização:** O jeito mais simples é plotar os dados em gráficos box plots são ótimos pra isso porque mostram claramente os quartis e marcam os pontos que estão muito fora da distribuição normal scatter plots também ajudam a ver outliers em relação a duas variáveis

**Z-Score:** Essa técnica calcula quantos desvios padrão cada ponto está da média geralmente pontos com Z-score maior que 3 ou menor que -3 são considerados outliers é bem útil quando seus dados seguem uma distribuição normal

**Isolation Forest:** É um algoritmo de machine learning específico pra detectar outliers a ideia é que outliers são mais fáceis de isolar do que pontos normais então o algoritmo cria árvores de decisão aleatórias e vê quais pontos são isolados mais rapidamente

## Conclusões

Esse card foi bem completo e cobriu vários aspectos importantes de trabalhar com dados do mundo real

O KNN mostrou como algoritmos simples podem ser poderosos pra classificação e sistemas de recomendação só precisa entender bem a escolha de K e da métrica de distância

O PCA foi meio surpreendente porque você consegue jogar fora dimensões mas manter quase toda a informação importante o que é essencial quando você tá lidando com datasets grandes ou precisa visualizar dados de alta dimensionalidade

O Q-Learning abriu minha cabeça pra aprendizado por reforço que é bem diferente do supervisionado mas super útil pra situações onde você não tem exemplos rotulados mas tem um objetivo claro e pode dar recompensas e punições

O K-Fold Cross-Validation é essencial pra ter certeza que seu modelo realmente funciona e não tá só decorando os dados de treino é um daqueles conceitos que parece meio óbvio depois que você entende mas faz toda diferença na prática

Outliers e data cleaning em geral mostraram que a parte menos glamourosa do data science que limpar dados é na verdade uma das mais importantes você pode ter o algoritmo mais sofisticado do mundo mas se seus dados são uma bagunça o resultado vai ser ruim

No geral ficou claro que trabalhar com dados reais é bem mais complicado que os exemplos de tutorial que a gente vê mas também é mais interessante porque você vê aplicações práticas de tudo o importante é ter uma boa compreensão dos fundamentos e saber quando usar cada técnica

## Referencias

Vídeo do card: 10 - Prática: Lidando com Dados do Mundo Real (II) e da pasta que esta no card

A imagen do Pack Man foi tirada do site: https://www.berkeley.edu/