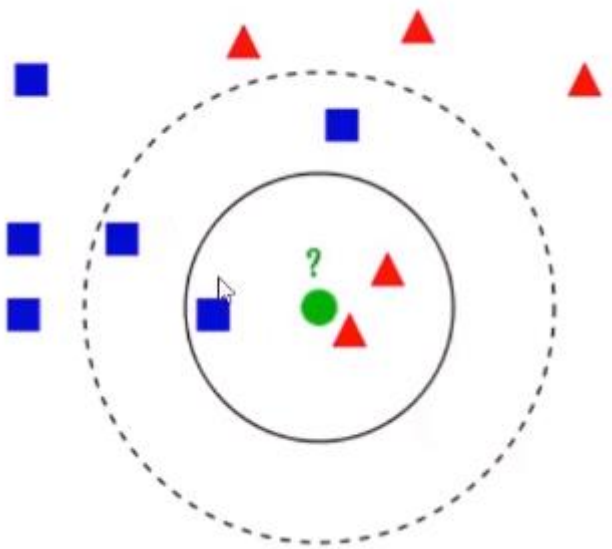
K-Nearest Neighbors (KNN)

Técnica usada para classificar um novo dado plotado baseando-se na classificação dos K vizinhos (pontos já plotados e classificados).

Exemplo: suponha que a imagem abaixo represente uma região de um scatterplot de “Avaliações” x “Popularidade” de filmes, e que cada ponto seja um filme. Os quadrados azuis são filmes de ficção científica e os triângulos vermelhos são de drama. O ponto verde é um novo filme que ainda não foi classificado por gênero.



Se utilizarmos o Método KNN com o argumento K=3, então a conclusão seria de que o filme é de drama, já que há 2 triângulos vermelhos contra 1 quadrado azul na amostra capturada.

Já se considerarmos K=5, classificaremos o filme como sendo de ficção científica, já que há 3 quadrados azuis contra 1 triângulo vermelho na amostra.

A escolha do valor de K é importante para não capturar uma amostra pequena demais, que pode estar sujeita a distorções por valores discrepantes, mas também não capturar pontos muito distantes que podem ter pouca relação com o novo dado, diminuindo o peso de dados mais relevantes.

Normalmente são aplicados diferentes valores de K para classificar diferentes dados de treino, e o valor que classificou os dados com maior precisão é o escolhido para classificar novos dados.

Este é um método usado em Aprendizado de Máquina Supervisionado, já que se usa dados conhecidos e rotulados por humanos para classificar um novo dado.

Redução de dimensionalidade

Dimensões num contexto de ciência de dados são diferentes atributos de um objeto ou problema. Por exemplo, uma flor tem uma cor para as pétalas, uma para o caule, tem uma altura, uma certa quantidade de pétalas etc. Todas essas características/atributos podem ser vistas como as dimensões de uma flor. Mas quanto mais dimensões, mais difícil é de representá-las em Data Frames e, principalmente, em gráficos. Além disso, menos dimensões significa menos dados, por isso é um conceito muito aplicado à compressão de arquivos, como imagens.

Existem técnicas utilizadas para diminuir as dimensões de um problema ou objeto mantendo sua variância o mais inalterada possível. Uma delas é a Principal Component Analysis (PCA).

Principal Component Analysis (PCA)

Consiste em encontrar hiperplanos em um dataset de muitas dimensões e projetar os dados neles. Esses hiperplanos são definidos por autovetores (eigenvectors), que são os componentes principais que dão nome à técnica. Encontra-se os autovetores, define-se os hiperplanos e projeta-se os dados neles, preservando a maior parte da variância desses dados. O número de hiperplanos definidos será o novo número de dimensões do dataset.

Os autovetores são compostos por proporções diferentes dos atributos originais, de acordo com o quanto de variância cada um desses atributos dá (quanto mais, maior é a porcentagem do autovetor determinada pelo atributo).

A ideia principal é descobrir e separar os padrões/informações mais importantes, descartando informações menos impactantes para a classificação dos elementos da amostra (as que geram menos variância) e assim podendo-se diminuir o número de dimensões.

Reduzir a dimensionalidade é muito útil para a representação visual dos dados e compressão de arquivos, além de deixar mais claro quais atributos originais contribuem mais para a variância dos dados.

Data Warehousing

Um conjunto de técnicas e processos usados para normalizar e esquematizar dados vindos de diversas fontes e formatos em um só sistema de banco de dados (a Warehouse), para que depois possam ser usadas em técnicas avançadas de ciências de dados.

Grandes empresas geralmente têm um departamento dedicado a isso, já que a requisição, normalização e estruturação desses dados são um processo que envolvem muitas e diversas técnicas e ferramentas. Desafios como lidar com dados incompletos ou corrompidos, a escolha de um sistema de consultas (SQL, Tableau etc), manutenção das consultas (já que URLs e permissões podem mudar), rotulação, detecção de bots, entre outros, são parte do trabalho de quem atua numa Data Warehouse.

Existem 2 grandes frameworks para realizar esses trabalhos: ETL e ELT.

Extract, Transform, Load (ETL)

O modo mais tradicional de fazer Data Warehouse. Consiste em extrair os dados da fonte (HTMLs, pesquisas, comentários etc), transformá-los para um esquema tabular que faça sentido para a Warehouse e depois carregá-los para ela. Mas esse framework pode se tornar problemático no passo de transformação dos dados quando a quantidade de dados começa a ficar grande demais.

Extract, Load, Transform (ELT)

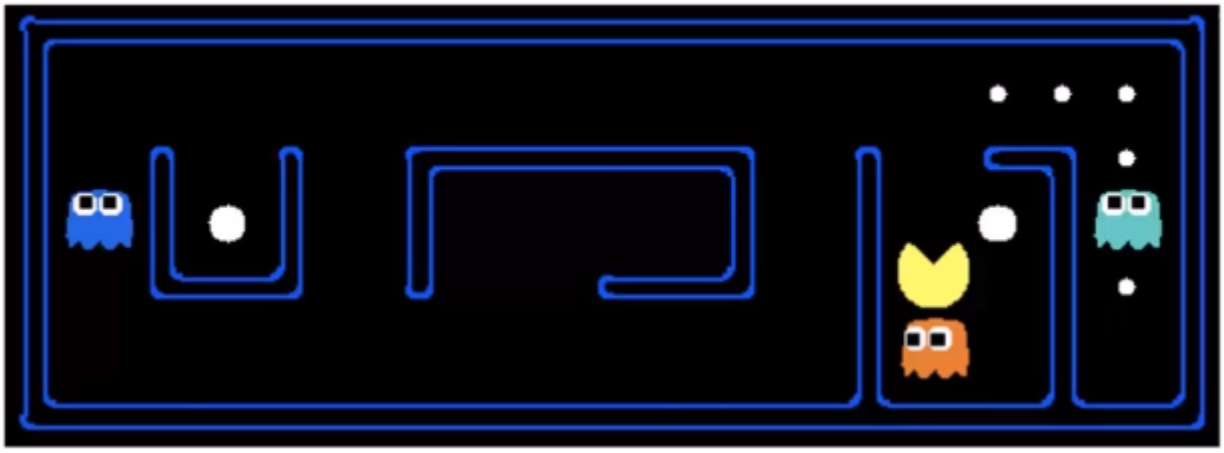
Conforme a internet foi se tornando maior e a quantidade e variedade de dados foi aumentando (e foram virando Big Data), foi ficando cada vez mais difícil para computadores e servidores transformarem os dados para serem armazenados de forma estruturada em bancos de dados. Para solucionar isso, inverteu-se os passos de armazenar e transformar os dados no sistema de banco de dados. Agora, é possível carregar os dados brutos para o banco para que eles sejam processados lá, onde geralmente se tem mais poder computacional. Para tornar isso ainda mais eficiente, ecossistemas como o Hadoop surgiram, um conjunto de aplicações que usam uma lógica de dividir e conquistar com os dados, repartindo-os em clusters e esquematizando-os em diferentes

máquinas para depois juntá-los de novo. Um processamento paralelo, com vários hardwares trabalhando para processar o conjunto de dados de forma mais eficiente, com divisão de trabalho.

Aprendizado por reforço

Tipo de aprendizado de máquina que consiste em deixar um agente explorar um ambiente e ir assinalando valores para cada ação (mudança de estado) tomada, conforme resultados positivos e negativos, assim aprendendo a realizar melhores ações no futuro.

Por exemplo, nessa situação o Pac-Man tem um fantasma ao sul, uma parede a oeste, nada a leste e nada ao norte. Este seria o estado do Pac-Man.



Mover ao sul neste caso é algo ruim, então essa mudança de estado teria um valor negativo assinalado a ela. Mover a oeste ou ao norte é neutro, então um valor nulo seria adicionado a essas mudanças de estado. Mover a oeste simplesmente não é uma opção aqui.

Dentro de muitas iterações no treinamento do agente, ele se depararia com esse estado diversas vezes e passaria a agir conforme o valor associado a cada mudança de estado anteriormente, escolhendo o de maior valor na maioria das vezes.

Q-Learning

Uma implementação específica do Aprendizado por Reforço.

Nele se tem: um conjunto de todos os estados possíveis do agente no ambiente (s); um conjunto de ações possíveis em cada estado (a); um valor Q para cara par (s, a).

O programa começa com todos os Q zerados. O agente começa a explorar o ambiente e, conforme coisas boas acontecem logo após uma ação a num estado s, Q(s, a) aumenta, se algo ruim acontece, Q(s,a) diminui.

É possível também adicionar um “discount factor”, que aumenta o valor de mudanças de estado anteriores se elas levaram a uma de valor positivo, ou diminui se levaram a uma de valor negativo. Por exemplo, no caso do Pac-Man acima, indo a leste e depois a norte, ele come uma Power Pill, então o aumento de valor na mudança de estado que fez ele chegar na Power Pill (a norte) gera um aumento no valor da mudança de estado anterior (a leste). Na Q-Learning, esse discount factor se dá por:

Q(s, a) += discount * (reward(s, a) + max(Q(s’)) - Q(s, a))

Onde s e a são o estado e ação anterior, e s’ é o estado atual.

Mas também não é interessante que o Pac-Man siga sempre o caminho de maior valor, porque é possível que um outro caminho não explorado tenha um valor maior, ou mesmo que um de menor valor registrado leve a outro de alto valor dada uma sequência de mudanças de estado ainda não tentada. Para levar isso em conta, é adicionado um elemento de aleatoriedade na tomada de decisão do agente, onde na maioria das vezes ele realiza a mudança de estado de maior valor Q registrado, mas às vezes toma uma decisão aleatória, em nome da exploração do espaço.

Conceitos parecidos

O Markov Decision Process (MDP), que é um modelo matemático, descreve um algoritmo de aprendizado muito semelhante ao descrito acima, mas com mais exatidão e formalismo.

Programação dinâmica também segue uma lógica muito, onde os valores Q seriam subprogramas criados para resolver subproblemas de um problema maior e, quando esse subproblema é enfrentado de novo, seja no mesmo problema ou em outro, o subprograma é reutilizado.

Matriz de Confusão

Vamos supor que haja um teste para uma doença muito rara, que afeta apenas 0.01% da população. Este teste teria uma precisão de 99.9% apenas retornando “não” todas as vezes. Uma situação como essa demonstra como às vezes medir a precisão de um modelo não é o suficiente. Nesses casos, é comum se usar matrizes de confusão, que relacionam falsos positivos, verdadeiros positivos, falsos negativos e verdadeiros negativos.

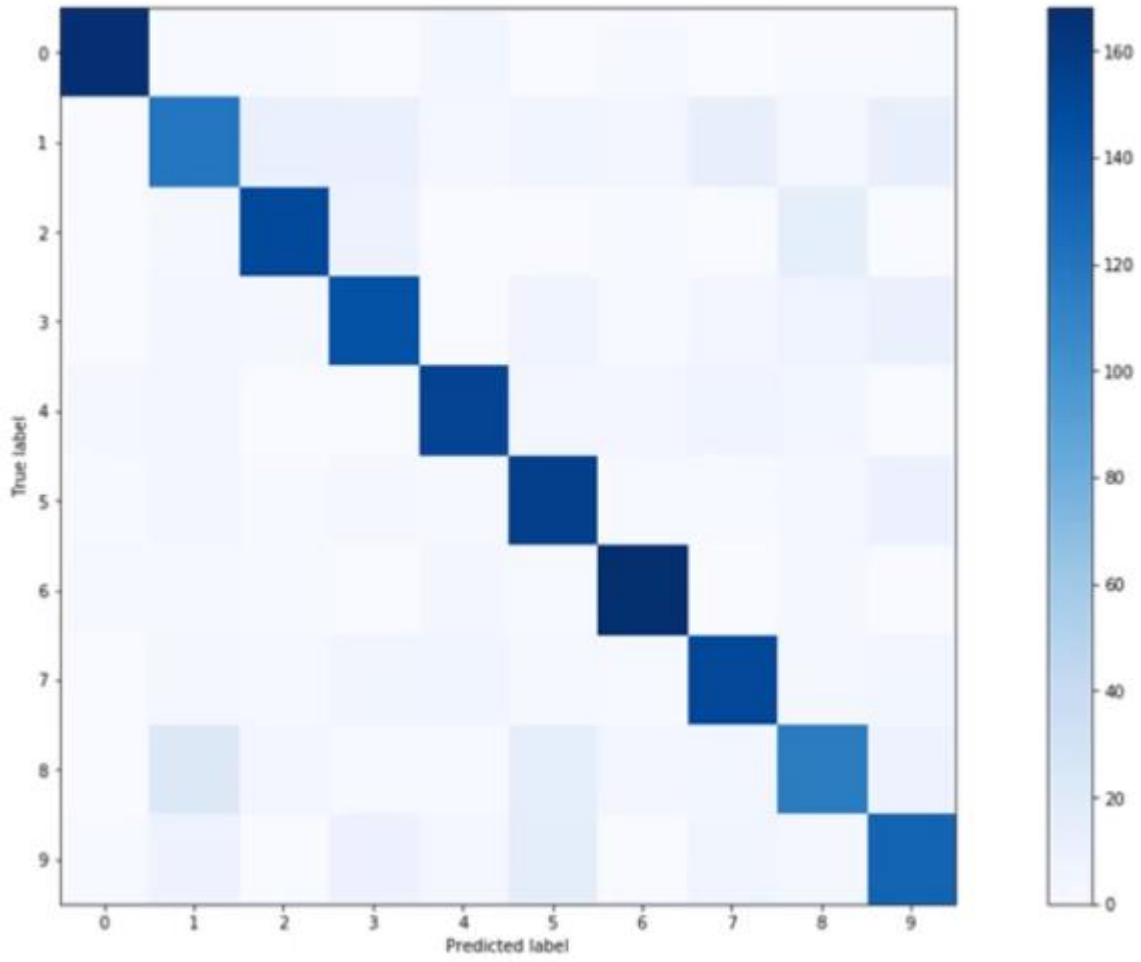
Exemplo de uma matriz de confusão binária genérica:

	Actual YES	Actual NO
Predicted YES	TRUE POSITIVES	FALSE POSITIVES
Predicted NO	FALSE NEGATIVES	TRUE NEGATIVE

Para um exemplo mais contextualizado, imagine uma IA que deve reconhecer se há um gato em uma foto. A matriz de confusão para essa IA seria assim:

	Actual cat	Actual not cat
Predicted cat	50	5
Predicted not cat	10	100

Um exemplo um pouco mais complexo seria a representação da precisão de uma IA que reconhece dígitos manuscritos através de uma matriz de confusão. A frequência de cada categoria da matriz é dada pela intensidade da cor, e os falsos positivos são mais específicos.



Recall

Uma métrica que se pode derivar de uma matriz de confusão é o Recall. É a porcentagem de positivos corretamente previstos. Uma métrica bem interessante quando os falsos negativos são de maior importância/problema (como em detecção de fraude).

$$\frac{TRUEPOSITIVES}{TRUEPOSITIVES + FALSENEGATIVES}$$

Precisão

Porcentagem de resultados relevantes. Boa métrica para quando falsos positivos são um problema muito grande (como exames médicos e testes de drogas)

$$\frac{TRUEPOSITIVES}{TRUEPOSITIVES + FALSEPOSITIVES}$$

Especificidade

O quão bom o modelo é em prever resultados verdadeiramente negativos.

$$\frac{TRUENEGATIVES}{TRUENEGATIVES + FALSEPOSITIVES}$$

F1 Score

Média harmônica da Precisão e Recall, ou seja, um meio termo das duas métricas.

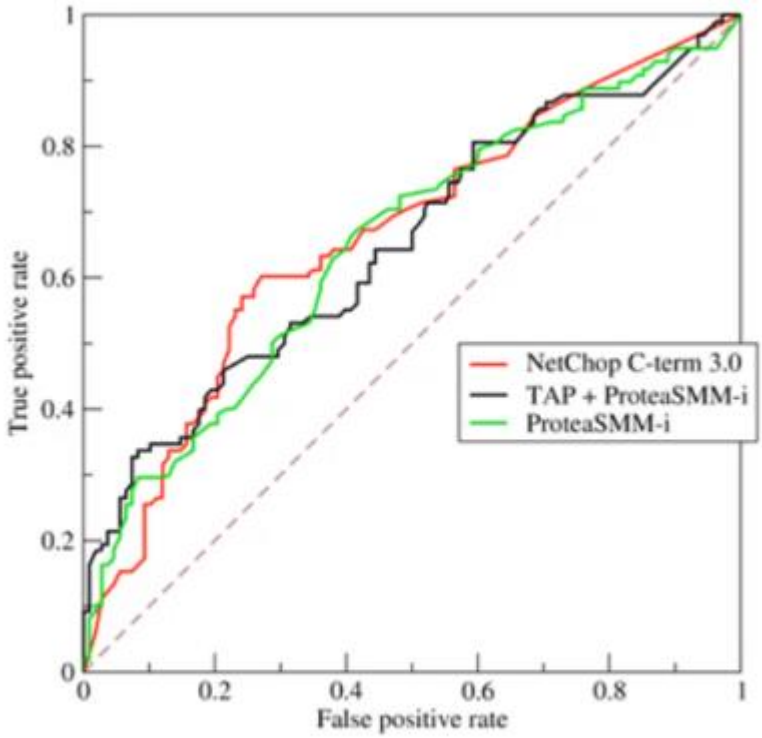
$$\frac{2 \cdot TRUEPOSITIVES}{2 \cdot TRUEPOSITIVES + FALSEPOSITIVES + FALSENEGATIVES}$$

Ou:

$$2 \cdot \frac{PRECISION \cdot RECALL}{PRECISION + RECALL}$$

Curva ROC

Receiver Operating Characteristic Curve. É a plotagem do Recall x Verdadeiros Falsos. Quanto mais curva para a esquerda, melhor. A linha na diagonal representa resultados aleatórios.



AUC

Area Under the Curve. Pode-se interpretar a área sob a Curva ROC como sendo a probabilidade de que o classificador sendo testado vá rankear uma medição positiva aleatória acima de uma negativa aleatória, o que seria o correto dado que a taxa de verdadeiros positivos está nas ordenadas. Uma AUC <= 0.5 indica que o classificador é inútil. Um perfeito teria uma AUC de 1.

Troca entre viés e variância

As predições de um modelo podem ser analisadas de acordo com seu viés e variância em relação à(s) resposta(s) correta(s).

O viés é o quão afastada da(s) resposta(s) correta(s) a média das suas previsões está.

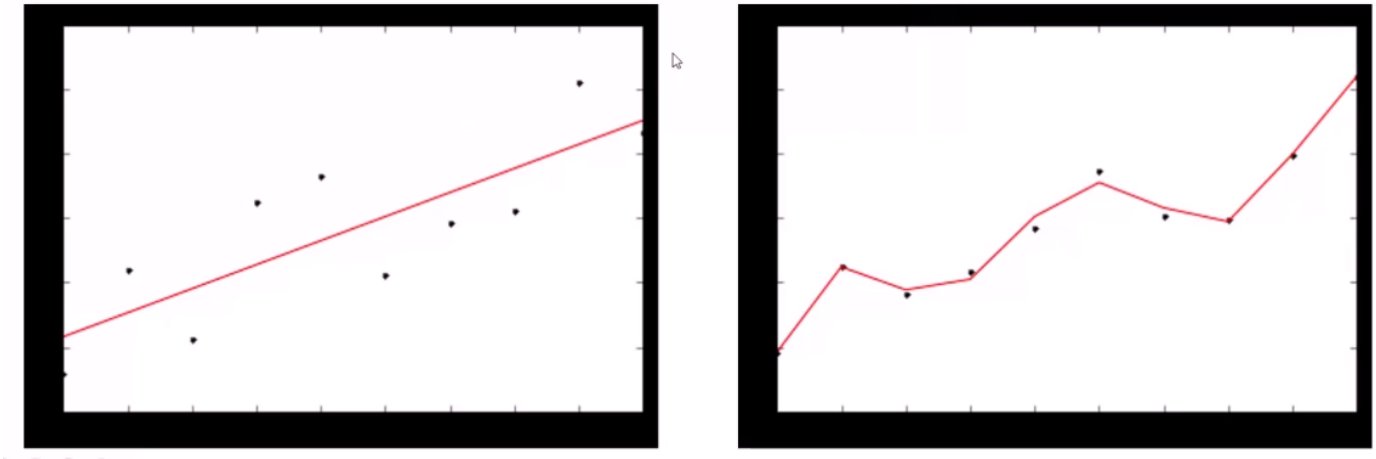
A variância é o quão dispersas estão as previsões estão da(s) resposta(s) correta(s).

Se considerarmos que respostas corretas são aquelas no centro do alvo e as previsões de um modelo são os furos, então:

- no alvo superior esquerdo: temos um modelo com alta variância e baixo viés (a média tende a ser no centro do alvo, porém as previsões estão dispersas, nenhuma de fato no centro do alvo);
- no alvo superior direito: temos um modelo com baixa variância e alto viés (a está bem deslocada do centro, porém as previsões estão bem aglutinadas);
- no alvo inferior esquerdo: temos um modelo com alta variância e alto viés, o pior caso possível;
- e, no alvo inferior direito: temos um modelo com baixa variância e baixo viés, o melhor cenário possível

A Troca

Na prática, o programador deve escolher se prefere ter menos variância ou menos viés em seu modelo, muito dificilmente fenômenos reais vão poder ser descritos por um modelo com ambas as métricas baixas.



Nessa imagem, temos à esquerda um exemplo genérico da plotagem de um modelo com baixa variância, mas alto viés (chamado de underfitting, ou seja, o modelo é pouco dobrado e sofisticado para caber nos dados), e à direita um com baixo viés, mas alta variância (chamado de overfitting, ou seja, dobrando e complexando o modelo ao máximo para caber nos dados).

Erro

No final do dia, o que realmente interessa é diminuir o erro. Trocamos variância por víes e vice-versa para tentar encontrar um ponto de equilíbrio onde o erro do modelo é mínimo. O erro pode ser dado em função da variância e viés pela equação:

$$error = bias^2 + variance$$

Um modelo muito complexo no melhor dos casos pode dar baixo viés e alta variância, já um modelo muito simples no melhor dos casos pode dar baixa variância e alto viés. O melhor modelo para minimizar o erro está entre esses dois casos num continuum de complexidade.

Por exemplo, no caso da técnica de K-Neareast-Neighbors, aumentando-se K se tem mais pontos de valores variados/espalhados em torno da resposta correta que tendem a “se cancelar” no cálculo da variância, diminuindo-a, porém também acaba-se incluindo pontos menos relevantes para a previsão que se está tentando fazer (pontos mais longe daquele que se está tentando inferir algo), levando a média das previsões para longe da resposta correta, aumentando assim o viés.

K-Fold Cross Validation

Uma técnica muito usada para evitar overfitting na modelagem.

Dado um modelo e um conjunto de dados, segmenta-se os dados em K partições, escolhe uma para ser a partição de teste e as outras K-1 para serem as de treino.

Após treinar o modelo com as K-1 partições e testar com a restante, salvar o coeficiente de determinação (R^2) do modelo para aquela partição, repete-se todos o processo para todas as outras K-1 partições.

No final, calcula-se a média dos R^2 encontrados e esse é tido como o do modelo final.

Essa maior variedade de divisão treino/teste do conjunto de dados previne que o modelo fique overfitted para 1 só par.

Limpeza e normalização de dados

Boa parte do trabalho de um cientista de dados é limpar e preparar seus dados. Os vem de diversas fontes e em diversos formatos e simplesmente alimentá-los aos modelos não é uma boa ideia, pois modelos treinados com dados ruins farão previsões ruins. Tipos comuns de “sujeira” que se pode encontrar em dados recém adquiridos:

- outliers (pontos discrepantes): que não colaboram em nada para o entendimento do fenômeno analisado e deformam as métricas;
- dados faltantes: lacunas vazias nos dados, informações incompletas. Há muitas formas de lidar com elas, seja descartando toda a seção da qual fazem parte ou preenchendo com algum item padrão ou função dos dados ao redor;
- dado maliciosos: dados deliberadamente manipulados e fabricados por pessoas tentando esconder algo ou passar uma informação errada para os modelos que os levaria a sugerir ações que as beneficiariam;
- dados errôneos: falhas em sensores ou problemas em APIs são alguns exemplos de falhas em processos que podem gerar dado incorretos
- dados irrelevantes: que não ajudam a descrever os fenômenos ou objetos nos quais você está interessado;
- dados inconsistentes: que representam a mesma informação, mas são escritos ou entregues de formas diferentes, como escrita ou ordem diferente;
- formatação: semelhante ao problema anterior, mas se referindo especificamente ao formato de escrita, como de datas ou pontuação.

É necessário sempre questionar a coerência de seus dados ir fazendo quantas correções forem necessárias, por tentativa e erro, até se ter um conjunto de dados que cumpre os requisitos de integridade para aquele contexto.

Normalização dos dados

Muitas vezes, o modelo sendo feito lidará com dados que representam quantidades muito diferentes, com escalas muito diferentes e que se relacionarão de alguma forma na análise (como idade e renda, ou altura e peso). Às vezes, isso será um problema, podendo gerar distorções nas análises. Isso pode ocorrer com um conjunto de dados de escala muito grande tendo mais peso que um com escala menor. Além de deformações aleatórias, os dados podem acabar enviesados para um grupo com mais dados de escalas maiores.

Para evitar esses problemas, é necessário se fazer a normalização dos dados antes de usá-los ou processá-los. Isso significa aumentar ou (geralmente) diminuir os valores de conjunto de dados mantendo a proporção de cada ponto em relação aos outros, levando-se em conta atributos da distribuição de todos os dados (como amplitude) na hora de se alterar um ponto individual.

Há diversas maneiras de se normalizar dados. Na biblioteca Scikit-learn, por exemplo, a implementação da PCA tem um parâmetro chamado “whiten” que, quando assinalado para “True”, normaliza os dados sendo passados de argumento. Essa mesma biblioteca tem um módulo de pré-processamento dos dados com várias funções de normalização.

Lidando com outliers

Muitas vezes é necessário remover outliers de um conjunto de dados para se conseguir representar o elemento típico da daquele universo, mas nem sempre é o caso.

Por exemplo, se está se calculando a renda média de um país, não se pode simplesmente remover bilionários, mesmo que eles enviesem a métrica. Já no caso de se analisar o tráfego de um website, é necessário remover registros com comportamento fora da curva, pois tendem a representar bots.

Identificando outliers

Num conjunto de dados, é comum se utilizar do desvio padrão para a identificação de outliers.

Numa distribuição normal, normalmente trata-se valores além de 2 ou 3 desvios padrões acima ou abaixo da média como outliers.

Num box plot, outliers são definidos como os pontos para além (acima ou abaixo) de 1.5 vezes a Amplitude Interquartil do Quartil 2.

Outras medidas também existem, a escolha de qual utilizar cabe ao cientista de dados, variando de caso a caso.

Engenharia de Características

Uma característica (feature) de um tipo de dado significa uma dimensão na representação numérica desse dado. Essa dimensão a mais aumenta consideravelmente a complexidade de achar padrões no conjunto para se criar algum tipo de modelo. Além disso, a dispersão dos dados também é aumentada conforme mais dimensões são analisadas. Levando isso em conta, é importante tentar reduzir ao mínimo de dimensões (características) de um conjunto de dados antes de usá-lo para treinar ou ser explorado por um modelo de ML. A prática dessa redução é chamada de Engenharia de Características (Feature Engineering).

Muitas vezes, esse processo resume-se à expertise na área do conhecimento aos quais os dados pertencem, o conhecimento sobre quais atributos de um elemento importam para determinada análise e a remoção, manutenção ou alteração manual desse mesmo.

Relacionada à expertise, tem-se a tentativa e erro. Simplesmente ir alterando os atributos do conjunto de dados a alimentando-os a um modelo até que as previsões fiquem mais precisas.

Mas também existem técnicas mais abrangentes e algorítmicas para se reduzir a dimensionalidade dos dados, como Principal Component Analysis (PCA) e K-means clustering. Essas também tendem a ser mais rápidas e fáceis de se implementar.

Cabe ao programador/analista entender qual a melhor quantidade de dimensões para os dados e quais métodos utilizar para chegar nesse número.

Técnicas de imputação de dados faltantes

Parte importante da Engenharia de Características é lidar com dados faltantes. É provável que, em casos reais, o conjunto de dados ao qual se tem acesso venha com lacunas.

Substituição pela média

Uma das técnicas existentes para se lidar com isso é a de Substituição pela Média, na qual simplesmente se substitui o campo do dado faltante pela média da coluna. É importante que seja

da coluna, que se refere ao mesmo atributo, já que não faz sentido preencher o dado com a média de outros atributos.

Em Pandas, pode-se aplicar esse método com a função fillna, especificando no argumento que se quer preencher com a média.

Essa técnica é pouco utilizada, pois:

- Funciona no nível da coluna apenas, perde relações entre atributos diferentes;
- Não se pode aplicar se os dados da coluna são categóricos;
- Não é muito preciso

Exclusão (dropping)

Outra técnica possível e pouco utilizada é a exclusão de linhas com dados faltantes.

Em Pandas, pode ser aplicada pela função dropna.

Só é razoável utilizá-la se as linhas excluídas realmente tiverem pouco impacto nas métricas do conjunto total e não haver muito tempo. Fora isso, é melhor partir para outro método.

Imputando com machine learning

Geralmente, o ideal é ser feito é se utilizar um modelo gerado por Machine Learning para imputar os dados (que serão usados em outro modelo).

Uma técnica possível é a de K-Nearest Neighbors, em que se preenche os campos faltantas com a média dos dados dos K campos mais próximos (similares). Porém o ideal é se utilizar essa técnica apenas para dados numéricos.

Para dados categóricos, é muito comum se utilizar de Deep Learning, já que redes neurais funcionam muito bem com esse tipo de dado.

Também é possível se utilizar de regressões lineares e não lineares para encontrar relações entre atributos e assim preencher os dados faltantes em função dos atributos conhecidos. A técnica mais avançada é a MICE (Multiple Imputation by Chained Equations).

Encontrar mais dados

A maneira ideal de lidar com dados faltantes é simplesmente encontrando-os. Buscando fontes alternativas ou mudando o método de extração/requisição. A primeira opção a se considerar deve ser sempre essa e partir para aproximações apenas se realmente for muito custoso ou impossível encontrar esses dados.

Lidando com dados desbalanceados

Um conjunto de dados desbalanceado é um no qual há uma discrepância muito grande entre a quantidade de casos positivos e negativos.

Casos positivos são aqueles que o modelo está tentando detectar (num modelo de detecção de fraude, por exemplo, um dado fraudulento seria um caso positivo), e o negativo é quando o evento procurado não ocorre.

Seguindo o mesmo exemplo de um modelo de detecção de fraudes, se for treinado com dados em que 99.9% dos pontos são não fraudulentos (que tende a ser um conjunto normal, esperado na prática), o modelo não vai aprender direito os padrões que caracterizam uma fraude e, portanto,

não vai conseguir detectá-las. Mesmo assim, teria uma precisão de 99.9%, passando a impressão errada de que é um bom modelo.

Por outro lado, um conjunto de dados de treino com uma quantidade muito grande de casos positivos pode ficar enviesado a retornar positivo mesmo em casos negativos. No mesmo modelo que testa fraude, por exemplo, se fosse no contexto de transações bancárias, poderia tornar fazer uma transação um processo muito custoso, com muitas investigações e bloqueamentos desnecessários engatilhados, além de problemas jurídicos.

Oversampling

Uma solução comum para isso é o oversampling, que consiste simplesmente em pegar o subconjunto de dados minoritário e replicá-lo no conjunto várias vezes para aumentar sua representatividade nos padrões aprendidos pelo modelo.

Undersampling

Outro método que pode ser usado nesse contexto é o undersampling, que consiste em pegar o subconjunto majoritário e diminuí-lo. Porém, excluir dados é algo que deve ser evitado, pois informações e nuances são descartados. O único caso em que talvez essa seja a solução é quando há uma limitação de tempo em hardware e o oversampling simplesmente geraria um conjunto de dados de treino ou exploração maior do que o setup atual suporta.

SMOTE

Uma solução mais avançada e precisa é a Synthetic Minority Oversampling Technique (SMOTE), que consiste em gerar novas amostras dos casos minoritários a partir das existentes com KNN.

Ajustar Limites

Uma maneira de lidar com o caso de excesso de falsos positivos, é aumentando o limite/tolerância entre os casos positivos e negativos, ou seja, fazer uma mudança no modelo/algoritmo ao invés de no conjunto de dados. Isso garante menos falsos positivos, mas pode aumentar os falsos negativos. Por isso a proporção do aumento do limite deve ser bem pensada e testada.

A mesma lógica vale para o caso em que se tem muitos falsos negativos, porém agora diminuindo o limite/tolerância, deixando menos possibilidades de valores classificados como negativos. O possível problema é surgir um excesso de falsos negativos.

No caso de um modelo de detecção de fraude, por exemplo, o primeiro caminho tende a ser o preferível dentre esses dois. Varia de caso a caso.

Outras técnicas de Engenharia de Características

Binning

Classificar dados numéricos em categorias baseado na faixa de valor que esse dado cai.

Por exemplo, dado um conjunto de pessoas, é possível categorizá-las pela década da idade, colocando todo mundo que tem entre 20 e 29 anos em uma categoria, quem tem entre 30 e 39 em outras etc.

Uma maneira mais geral de se fazer isso é por quartil, ordenando e separando os dados em categorias de mesmo tamanho. No final, transforma-se dados numéricos em ordinais.

No final, o uso dessa técnica transforma dados numéricos em categóricos.

De forma geral, só é útil quando há incerteza na medição dos dados, utilizando as categorias para englobar os erros. Fora isso, são muito específicos os casos em que é vantojoso fazer isso, já que na prática informações são perdidas nessa transformação.

Transformações

Alterar a distribuição dos dados utilizando-se de uma função (ou seja, mantendo as posições relativas dos dados) pode ser vantajoso para se identificar certas relações ou facilitar cálculos.

Muitas vezes uma relação não explícita ou detectada numa distribuição logarítmica pode ficar clara numa representação linear daqueles dados.

Um exemplo de aplicação disso é no algoritmo de recomendações do YouTube, como revelado pela Google. O modelo é alimentado com a distribuição original dos dados ($f(x) = x$) em conjunto com a quadrática ($f(x) = x^2$) e a radical ($f(x) = raiz(x)$). Isso permite a análise das relações entre os dados com mais perspectivas, mais ângulos, sendo cada uma das distribuições mais vantajosa para a identificação de um certo tipo de relação entre dados.

Como demonstrado pelo exemplo acima, transformar os dados não significa necessariamente substituí-los. Muitas vezes, alimentar o modelo com distribuições alternativas em conjunto com a original pode fazer com que ele tire o máximo daquele conjunto em termos de aprendizagem.

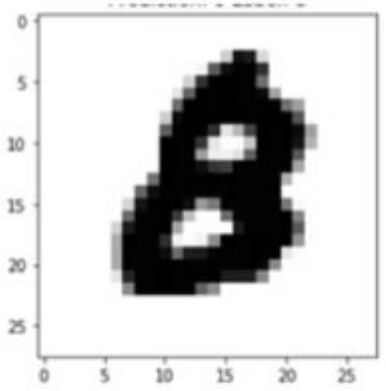
Encoding (codificação)

Representar os dados de um conjunto de uma maneira alternativa, através de alguma codificação, pode ser interessante ou até obrigatório para um modelo fazer uso deles. É uma maneira de fazer com que os dados tenham o significado desejado, que muitas vezes não é óbvio na forma original.

Um exemplo de codificação é o One-hot Encoding, que consiste em representar categorias por campos binários e dizer se aquela categoria corresponde a um certo dado associando a ele o conjunto de campos com 1 no que corresponde e 0 no que não corresponde.

Muito usado em redes neurais, já que neurônios só tem os estados de ativado e desativado.

Por exemplo:



Neste caso, cada campo binário indica a correspondência ou não do dígito manuscrito com o dígito representado pelo campo. Apenas o campo que representa o 8 está com o valor 1, indicando que o manuscrito acima é um 8.

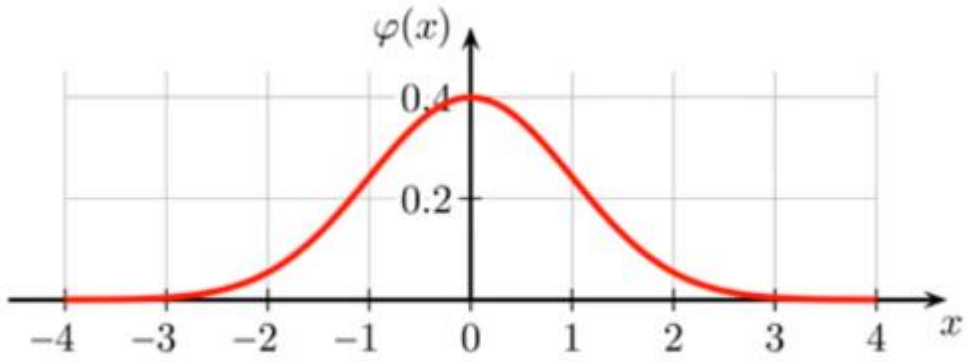
Dimensionamento e normalização

É importante que quando os dados representam ou possuem atributos com escalas muito diferentes seja feita a normalização, que consiste em mudar o valor absoluto dos dados mantendo a proporção e posição relativa entre eles.

A razão disso é que, se um certo atributo é representado por dados com valores muito grandes e outro por valores menores, o primeiro tende a ter um peso muito maior em como o modelo vai funcionar e, como são atributos diferentes, isso não faz sentido, já que um valor “grande” ou “pequeno” depende do que está sendo medido.

Um exemplo é quando se está modelando algo relacionado a pessoas, que possuem idades e rendas mensais. Uma idade de 80 é grande, porém, pensando em renda mensal em dólares, é pequena. Por isso, é importante normalizar esses dados para que ocupem a mesma escala e possam ser comparados e combinados sem distorções.

Muitos tipos de modelo preferem dados distribuídos normalmente com média 0, como no caso das redes neurais.



Mas existem muitos outros métodos de normalização. A biblioteca Scikit Learn tem um módulo pré-processador de dados com diversas funções de normalização.

Para finalizar, se a predição do modelo for um dado numérico, é importante inverter a normalização, escalar os dados para as dimensões originais, na hora de apresentá-los (após o processamento pelo modelo).

Embaralhamento

Muitas vezes é importante modificar a ordem dos dados aleatoriamente para eliminar qualquer influência do método de medição ou requisição, que pode entregá-los em um ordenamento específico que não é interessante para a análise.