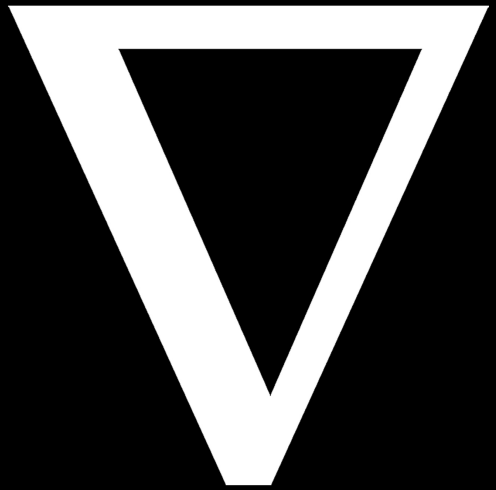


Técnicas de Ciência de Dados

João Marcos Cardoso da Silva
@joaomarcoscsilva

Scaling

Valores Numéricos

Para valores numéricos, não podemos simplesmente passar o número como entrada para o modelo preditor.

Isso porque se uma das informações estiver em uma escala maior que as outras, alguns modelos (como o KNN) podem não funcionar bem.

Escalonamento em um intervalo

Uma maneira de combater o problema das escalas distintas é fazer todas as features ficarem dentro de um mesmo intervalo (por exemplo, entre 0 e 1).

Para fazer isso, basta realizar a seguinte operação:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Problemas com o escalonamento

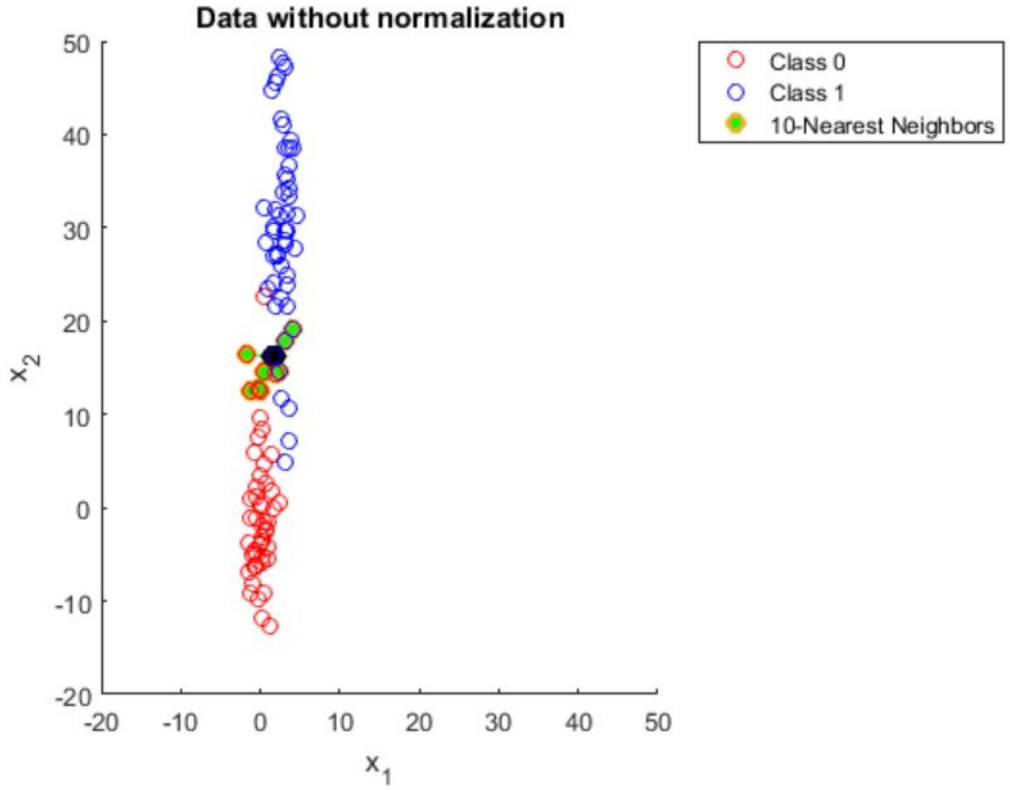
Por usar os valores máximos e mínimos, o escalonamento é **extremamente** sensível a outliers: um único valor muito alto ou baixo vai alterar completamente a escala dos dados.

Normalização

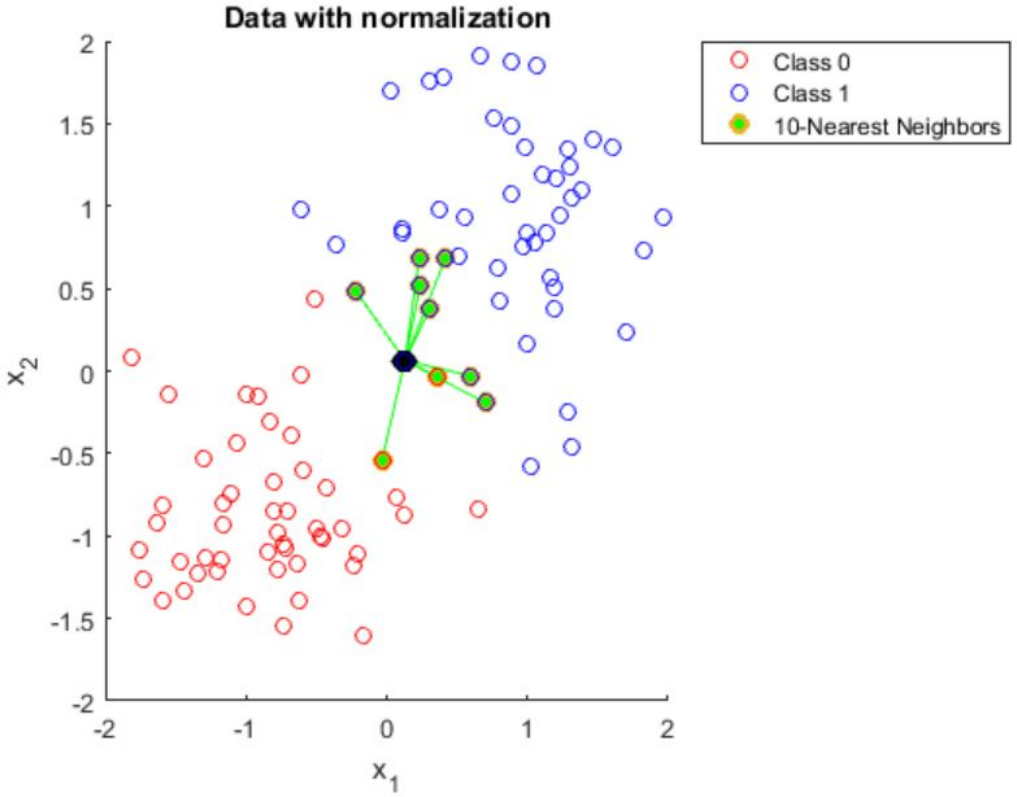
Uma alternativa a fazer o escalonamento é usar a normalização, que usa a média e o desvio padrão:

$$z_i = \frac{x_i - \mu}{\sigma}$$

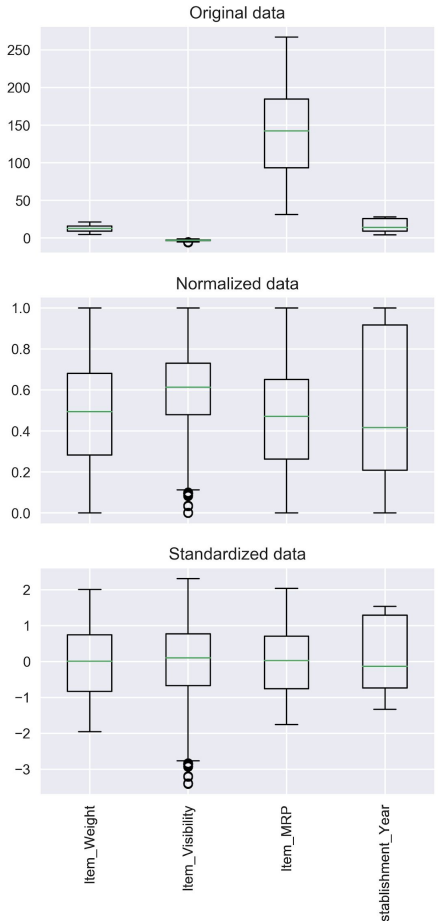
Isso garante que todas as features vão ter média 0 e desvio padrão 1, colocando elas aproximadamente na mesma escala.



Sem pré processamento, o KNN só vai considerar o eixo com maior escala!



Normalizando, o algoritmo pode usar as duas variáveis para a predição


Original data
250
200
150
100
50
0
Normalized data
1.0
0.8
0.6
0.4
0.2
0.0
Standardized data
2
1
0
−1
−2
−3
Item_Weight
Item_Visibility
Item_MRP
stablishment_Year

Encoding

Dados Categóricos

Dados categóricos são aqueles que só podem assumir um número finito de valores.

Exemplos:

- Gênero
- Tipo Sanguíneo
- Palavras (em um vocabulário finito)

Variáveis Binárias

Variáveis binárias só podem assumir dois valores (por exemplo, se um paciente apresentou ou não algum sintoma).

Nós podemos usar os valores 0 e 1 para indicar o valor de uma variável binária para um modelo.

Alguns modelos também podem usar -1 e 1, mas normalmente isso não vai fazer uma diferença grande.

Variáveis Não Binárias

Variáveis não binárias podem assumir mais de dois valores (por exemplo, o tipo sanguíneo de uma pessoa).

O que não fazer

Não podemos simplesmente atribuir números inteiros para uma variável não binária (no exemplo do tipo sanguíneo, seria como decidir arbitrariamente que A = 0, B = 1, AB = 2, O = 3, etc.).

Fazendo isso, nós estaríamos definindo uma ordem para os valores (por exemplo, o tipo A seria “menor” que o tipo O) e também uma relação de proximidade (o tipo A está mais perto do B do que do O) o que normalmente não faz sentido, mas pode influenciar o modelo.

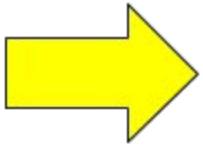
One-Hot Encoding

Em vez de usar valores inteiros, vamos usar vetores One-Hot.

Para isso, nós vamos representar cada valor como um vetor com um elemento pra cada possível categoria. Esse vetor vai ter um número 1 na entrada correspondente ao valor da variável e um número 0 em todas as outras.

One-Hot Encoding

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

“Tamanhos” das classes

Com One-Hot encoding, não temos uma variável “menor” que a outra, pois todos os vetores tem o mesmo tamanho (uma única entrada 1, com 0 nas demais)

“Distâncias” entre as classes

Com One-Hot Encoding, também temos a vantagem de que nenhum valor está “mais perto” do outro, já que a distância entre os vetores é sempre a mesma.

Desvantagens

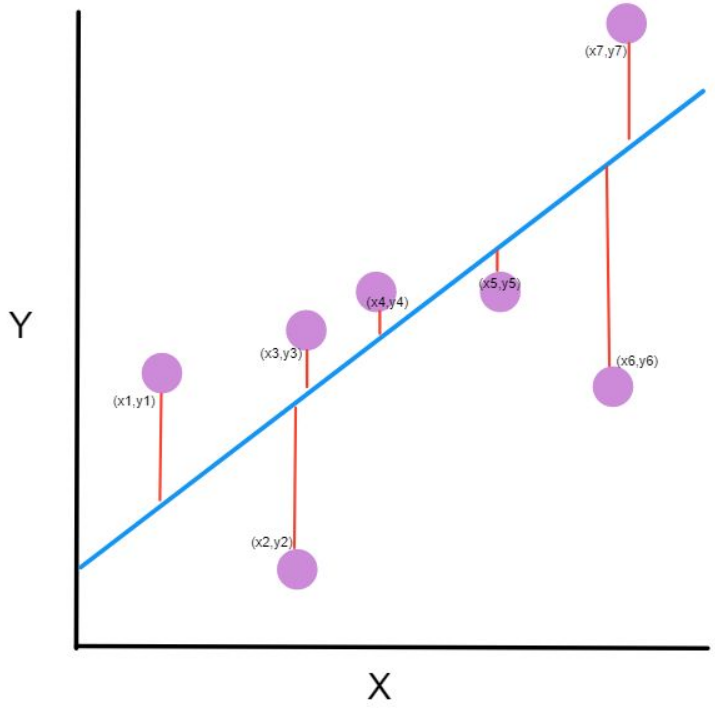
Uma desvantagem do One-Hot Encoding é que ele ocupa muito mais espaço na memória, pois precisa guardar, para cada variável categórica, uma coluna inteira para cada classe possível.

Por esse motivo, normalmente os dados salvos nos datasets normalmente não estão no formato One-Hot, então precisamos fazer essa alteração manualmente.

Métricas de Regressão

Métricas de Regressão

Para tarefas de regressão, o valor predito pelo modelo é um número real. Por isso, não podemos só medir a acurácia (quantos valores estão exatamente certos), já que um erro de 0.1% é preferível a um erro de 50% do valor.


(x7,y7)
(x5,y5)
(x4,y4)
(x3,y3)
(x6,y6)
(x1,y1)
(x2,y2)
Y
X

Erro Quadrático Médio

A métrica mais usada para regressão é o erro quadrático médio (MSE), que calcula a média do quadrado da diferença entre o valor predito e o correto:

$$\frac{1}{n}\sum(Y_i - \hat{Y}_i)^2$$

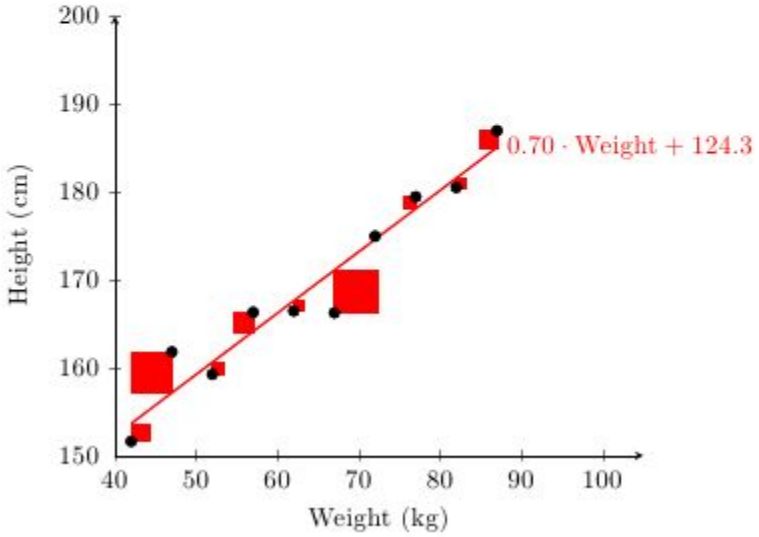
Elevando ao quadrado, a métrica penaliza mais os erros maiores que os menores.

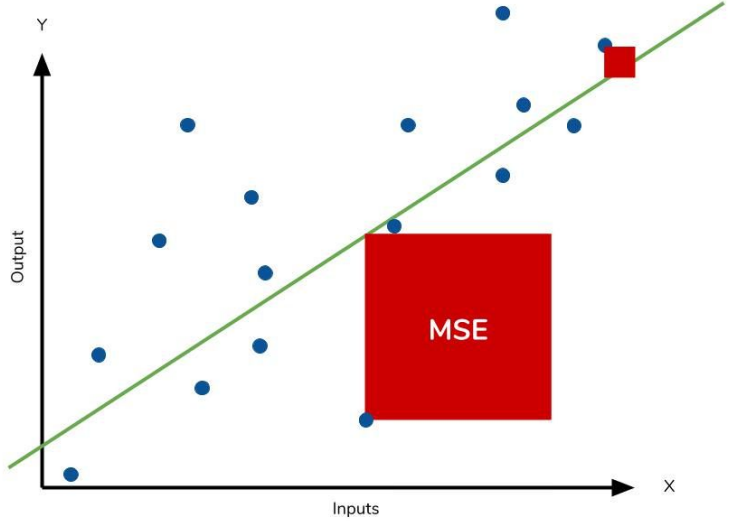
Raiz do Erro Quadrático Médio

Também podemos usar a raíz do MSE como uma medida do erro:

$$\sqrt{\frac{1}{n}\sum(Y_i - \hat{Y}_i)^2}$$

Usar a raiz tem a vantagem de ser mais fácil de interpretar, já que o resultado vai ter a mesma unidade do valor original.

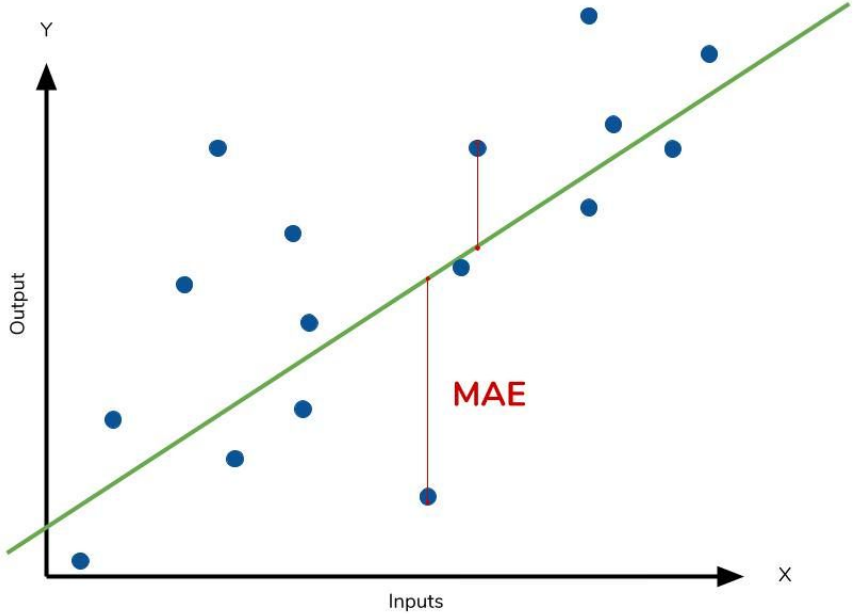

200
190
180
170
160
150
Height (cm)
40
50
60
70
80
90
100
Weight (kg)
0.70 · Weight + 124.3


Y
Output
MSE
X
Inputs

Erro Absoluto Médio

O erro quadrático médio tem a desvantagem de ser muito sensível a outliers (um único valor com um erro grande pode mudar muito o seu valor). Para combater isso, podemos usar o erro absoluto médio:

$$\frac{1}{n}\sum|Y_i - \hat{Y}_i|$$


Y
Output
MAE
X
Inputs

Problemas com métricas de regressão

Um possível problema com as métricas MSE ou RMSE é que não temos uma escala para comparação: o mesmo valor pro erro pode ser muito alto ou muito baixo dependendo se o problema é mais ou menos complexo.

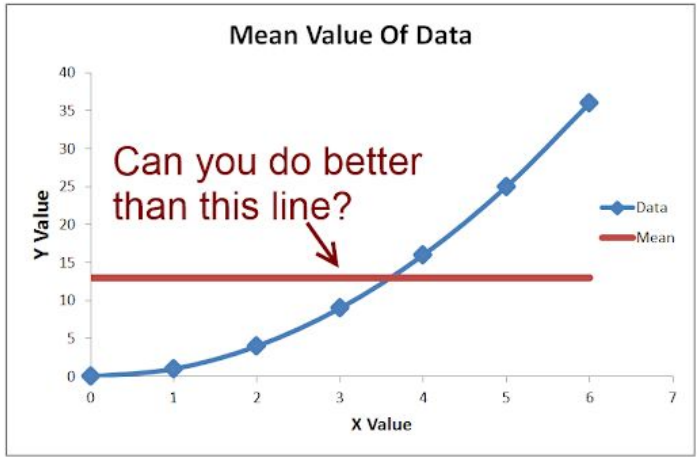
Construindo uma baseline

Para facilitar a interpretação, podemos comparar os resultados com um modelo extremamente simples: vamos ignorar a entrada e predizer sempre a média dos valores da saída:

$$f(x) = \mathbb{E}[Y]$$

A Baseline Constante

Esse modelo é extremamente fraco: ele faz a mesma predição para todos os valores na entrada! Então uma bom teste é verificar se o seu modelo mais complexo consegue, pelo menos, um erro menor que o da baseline:



O Erro da Baseline

O erro dessa baseline constante é igual ao valor da Variância da variável Y:

$$f(x) = \frac{1}{n}\sum(Y_i - \mathbb{E}[Y])^2 = \mathrm{Var}[Y]$$

Logo, o nosso erro MSE deveria ser **pelo menos** menor que a variância.

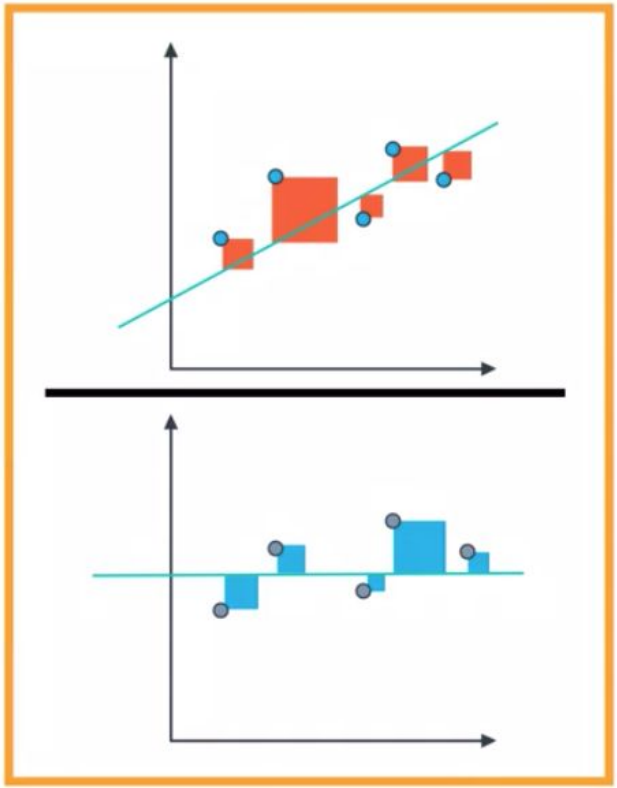
Coeficiente de Determinação (ou R^2)

A medida R^2 une o erro MSE do modelo com o da baseline em um único valor:

$$R^2 = 1 - \frac{\text{MSE}_{\text{modelo}}}{\text{MSE}_{\text{baseline}}} = 1 - \frac{\text{MSE}_{\text{modelo}}}{\text{Var}[Y]}$$

- Se $R^2 = 1$, o modelo é perfeito e tem erro 0
- Se $R^2 = 0$, o modelo é tão ruim quanto a baseline constante
- Se $R^2 < 0$, o modelo é pior que a baseline constante

$$R2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$



Métricas de Classificação

Métricas para Classificação

Para tarefas de classificação, a métrica mais comum é a acurácia, que mede a porcentagem de predições corretas feitas pelo modelo.

Problemas com a Acurácia

- Não funciona bem para dados desbalanceados (por exemplo, é fácil conseguir 99% de acurácia se queremos classificar uma doença que só afeta 1% da população)
- Considera que todos os erros têm o mesmo peso, o que pode não ser o caso (por exemplo, é muito melhor deixar um email spam passar pelo filtro do que perder a sua proposta de emprego)

Diferentes tipos de erros

Para classificação binária, existem dois tipos diferentes de erros, com diferentes importâncias dependendo do problema:

- Falso positivo
  - Predizer que uma pessoa saudável na verdade está doente
  - Predizer que um email normal na verdade é spam
- Falso negativo
  - Predizer que uma pessoa doente na verdade está saudável
  - Predizer que um email spam na verdade é normal

Falsos negativos e positivos

Não podemos simplesmente ignorar um tipo de erro e focar só no outro:

- Um modelo que diz que toda pessoa está doente tem 0 falsos negativos
- Um modelo que não diz que nenhum email é spam tem 0 falsos positivos

Em vez disso, normalmente precisamos encontrar um ponto de equilíbrio entre os dois

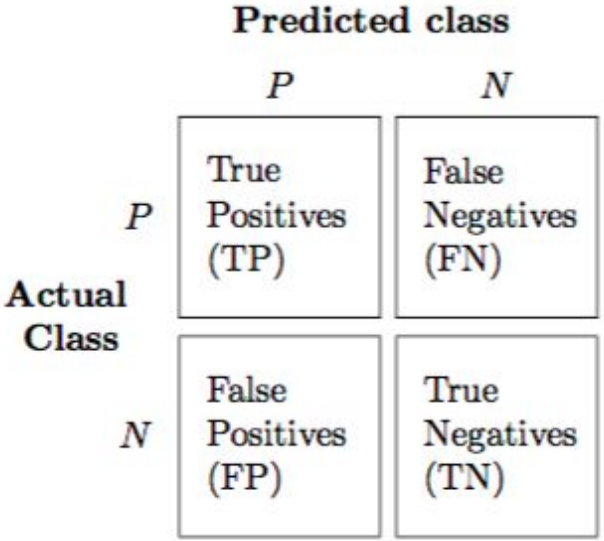
Matriz de Confusão

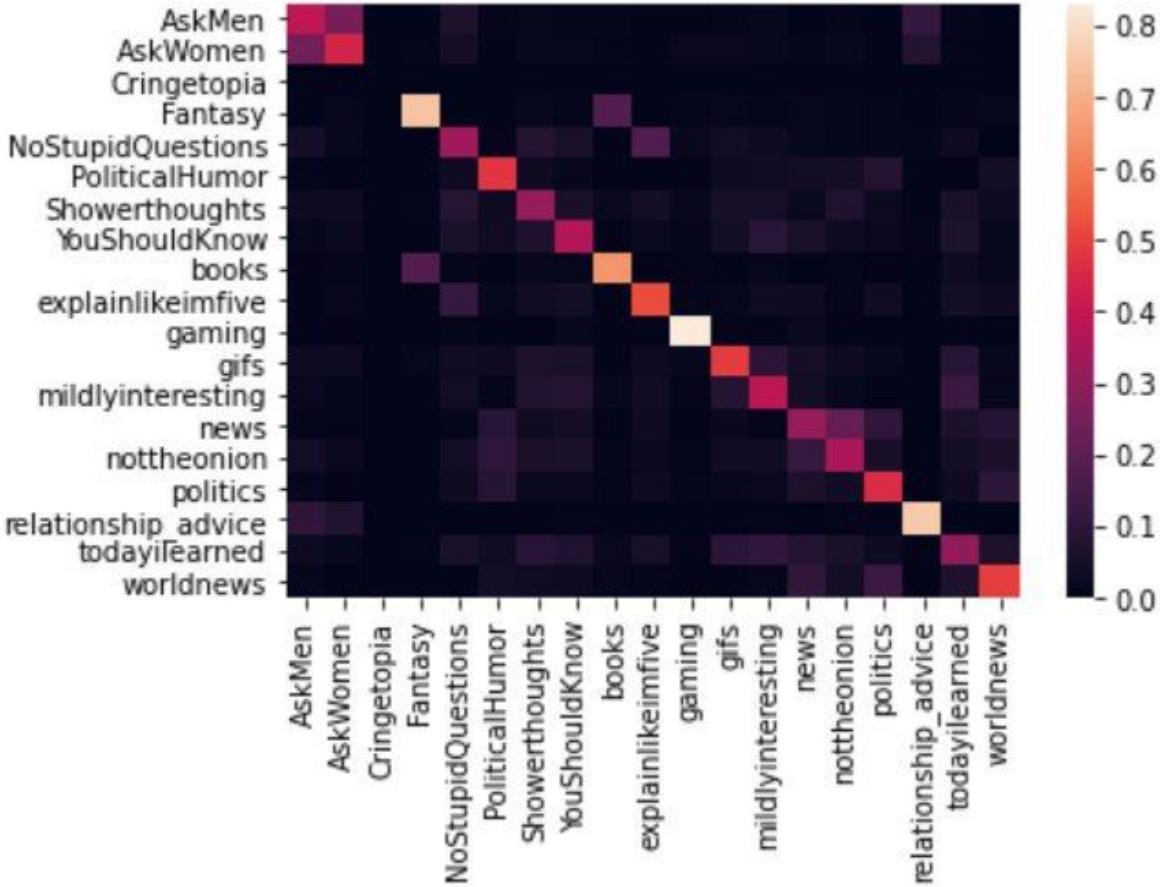
Também podemos estender esses conceitos para mais de duas classes, pensando, para cada classe, o quanto o modelo acerta e com quais classes ele se confunde.

Uma ferramenta extremamente poderosa é a matriz de confusão, que ajuda a indicar esses resultados de forma visual.

A matriz também pode ser muito útil para interpretar o funcionamento do modelo, indicando com quais classes ele tem mais dificuldade.

Exemplos


Predicted class
P
N
Actual Class
P
True Positives (TP)
False Negatives (FN)
N
False Positives (FP)
True Negatives (TN)


AskMen
AskWomen
Cringetopia
Fantasy
NoStupidQuestions
PoliticalHumor
Showerthoughts
YouShouldKnow
books
explainlikeimfive
gaming
gifs
mildlyinteresting
news
nottheonion
politics
relationship_advice
todayilearned
worldnews
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

Precisão

Podemos medir o quão precisas são as predições de um modelo com a seguinte métrica:

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Positivos}}$$

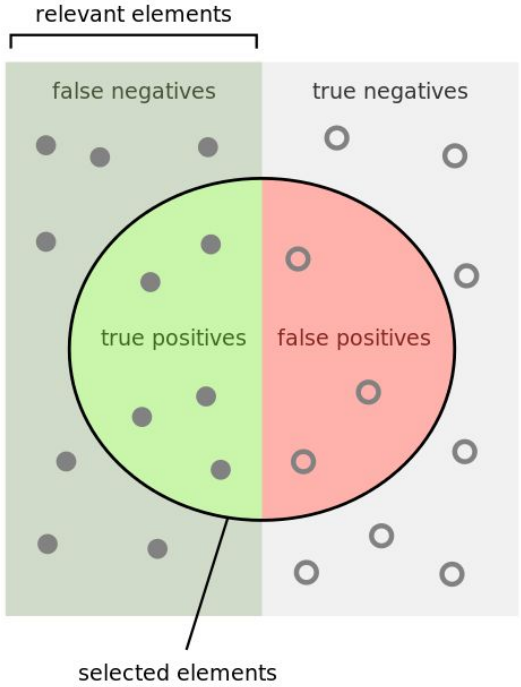
Em um modelo com alta precisão, uma predição positiva tem altas chances de ser realmente positiva.

Recall (Revogação ou Sensibilidade)

A precisão não considera os casos em que o modelo previu um resultado negativo, mesmo se o correto seria um positivo. Para medir isso, usamos a sensibilidade ou revocação:

$$\text{Recall} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Negativos}}$$

Em um modelo com alta sensibilidade, um resultado negativo tem grandes chances de ser verdadeiro.



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

Combinando as métricas

Assim como no caso dos falsos positivos e negativos, não podemos considerar **apenas** a precisão ou o recall isoladas, pois é fácil conseguir um modelo que vai bem em apenas uma delas mas é péssimo na outra.

Em vez disso, precisamos combinar a precisão e a sensibilidade de maneira que se qualquer uma das duas for muito baixa, o resultado final também será baixo.

Métrica F1

A métrica F1 é a média harmônica da precisão e do recall:

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}$$

Ela faz uma combinação das duas medidas de modo que se qualquer uma for muito baixa, o resultado final também será baixo.

Métrica F_β

Se quisermos dar um peso maior para o recall ou para a precisão, podemos usar a média harmônica ponderada, onde damos um peso de β^2 para o recall:

$$F_\beta = \frac{\beta^2+1}{\frac{\beta^2}{\text{recall}}+\frac{1}{\text{precision}}} = (1+\beta^2)\cdot\frac{\text{precision}\cdot\text{recall}}{(\beta^2\cdot\text{precision})+\text{recall}}$$

Métodos de classificação com probabilidades

Alguns métodos de classificação não predizem apenas uma classe, mas sim a sua probabilidade, como é o caso do Naive Bayes. Nesses algoritmos, nós podemos controlar a quais tipos de erros (falsos positivos ou negativos) nós queremos dar uma maior importância.

Convertendo probabilidades em Classificadores

A maneira mais simples de converter uma probabilidade em uma classificação é a seguinte:

- Se $p > 0.5$, classificamos como um exemplo positivo.
- Se $p < 0.5$, classificamos como um exemplo negativo.

Nesse caso, o valor 0.5 é o limiar que separa as decisões.

Variando o limiar de decisão

Para aplicações específicas, nós podemos usar limiares de decisão diferentes de 0.5.

Por exemplo, em um detector de spam, um falso positivo é muito mais sério que um falso negativo, então podemos classificar os exemplos como positivos só se tivermos **certeza** que uma mensagem é spam (por exemplo se $p > 0.9$)

Variando o limiar de decisão

Já em um exame médico, um falso positivo é menos sério que um falso negativo, então podemos classificar todas as pessoas como possivelmente infectadas com uma doença se $p > 0.2$, por exemplo.

Regras Gerais

Em geral, temos:

- Se falsos positivos são mais importantes, devemos usar um limiar pequeno
- Se todos os erros têm o mesmo peso, devemos usar um limiar de 0.5
- Se falsos negativos são mais importantes, devemos usar um limiar alto

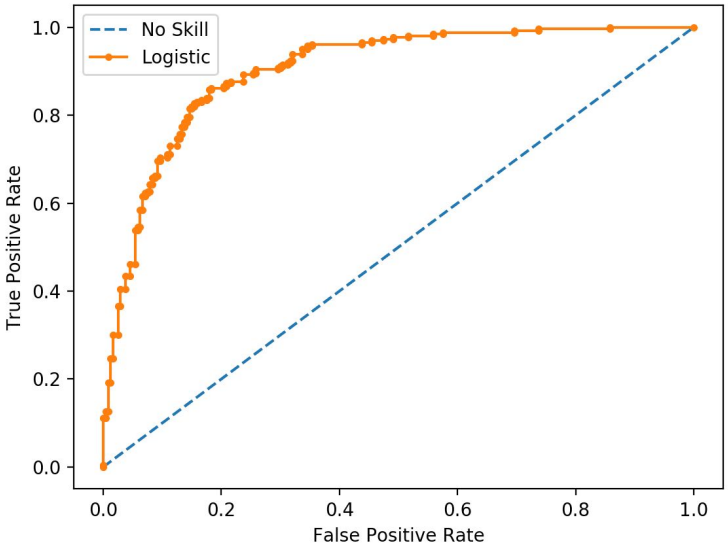
Limiares extremos

Se usarmos um limiar em 0, todos os itens serão classificados como positivos.

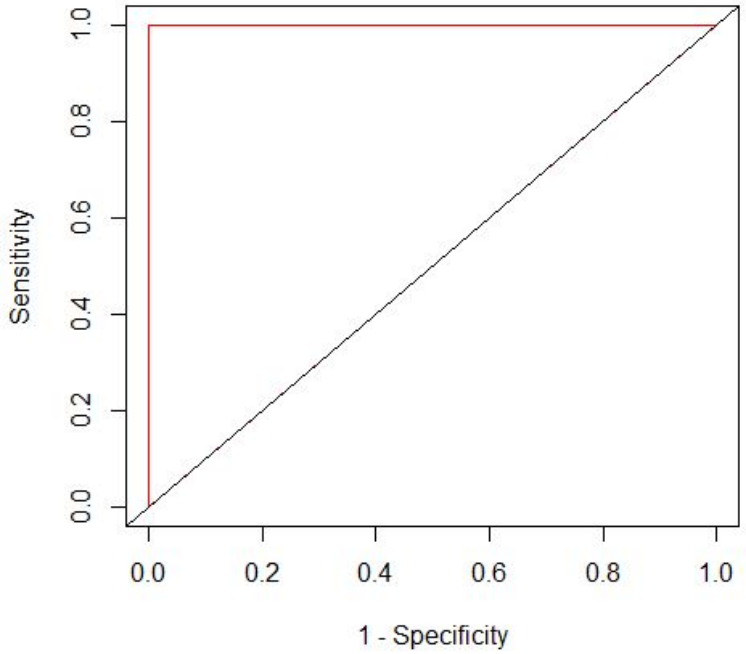
Se usarmos um limiar em 1, todos os itens serão classificados como negativos.

Curva ROC

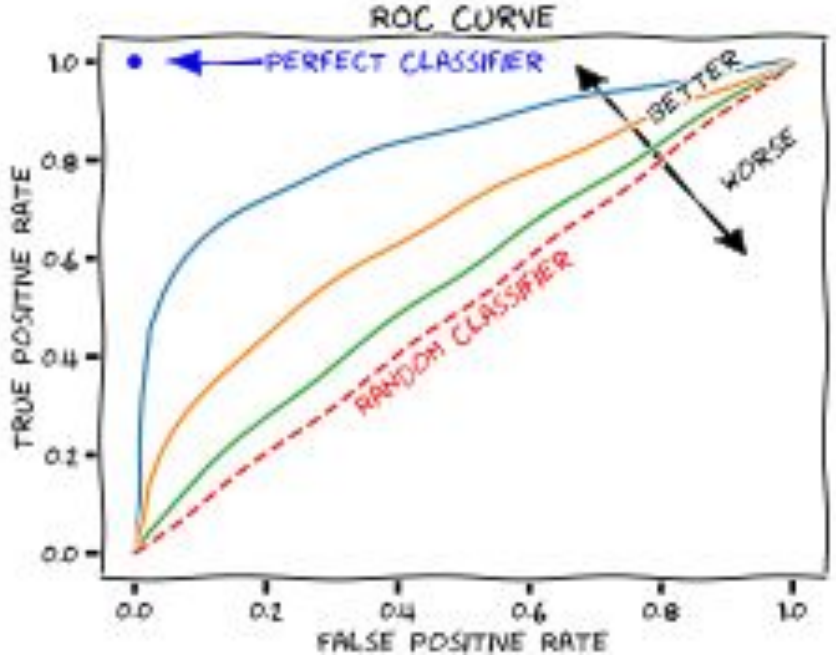
Uma maneira de visualizar o comportamento de um modelo é com uma curva ROC, onde nós visualizamos a taxa de falsos negativos e positivos para diferentes limiares:



Classificador perfeito

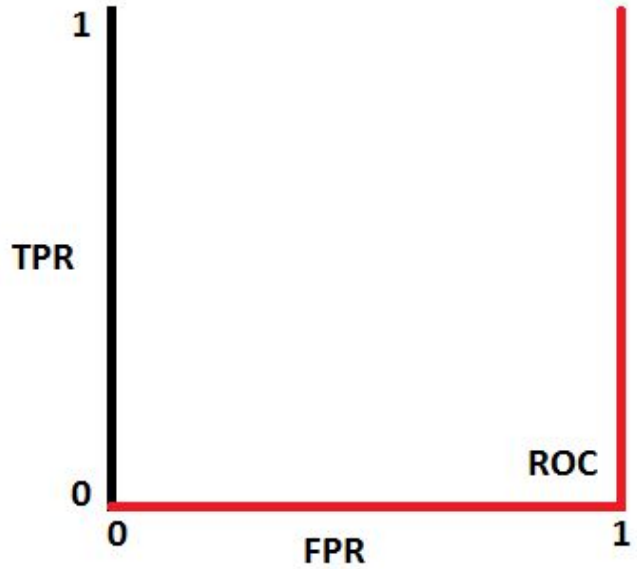


Classificador Aleatório



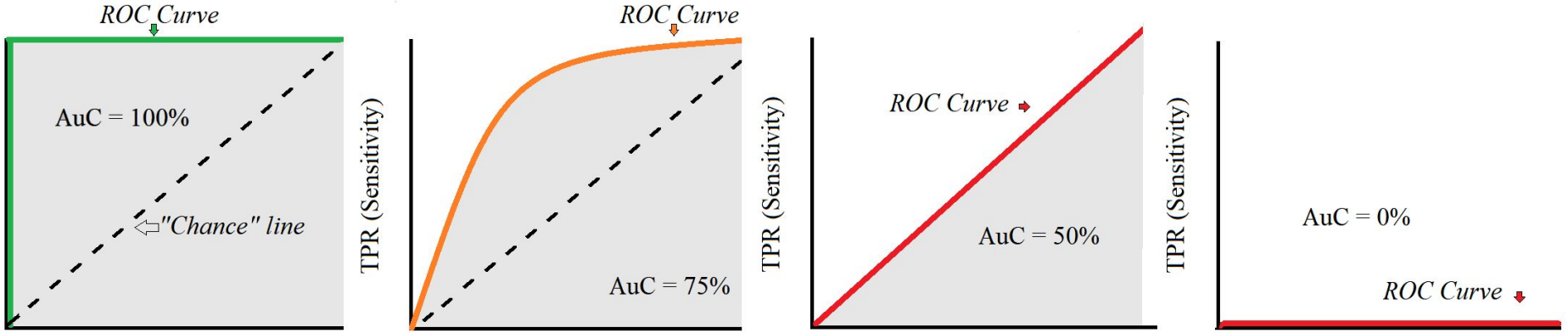
Pior classificador possível

O pior classificador possível é aquele que tem as predições trocadas (sempre que a saída é 0, ele prediz 1 e vice-versa). Por isso, sua curva é pior que a do aleatório:



AUC (Area Under Curve)

Uma possível métrica para classificadores é a AUC, que é a área total embaixo da curva ROC. Ela é sempre um valor entre 0 e 1, sendo 0.5 correspondente a um classificador aleatório



Avaliação de Modelos

Treino e Validação

Para avaliarmos um modelo, precisamos separar uma parte do Dataset para servir de validação.

Esses são dados onde nós conhecemos as respostas corretas, mas que não serão usados para treinar o modelo. Em vez disso, usaremos essas respostas para calcular a eficácia do modelo depois dele ser treinado.

Treino e Validação

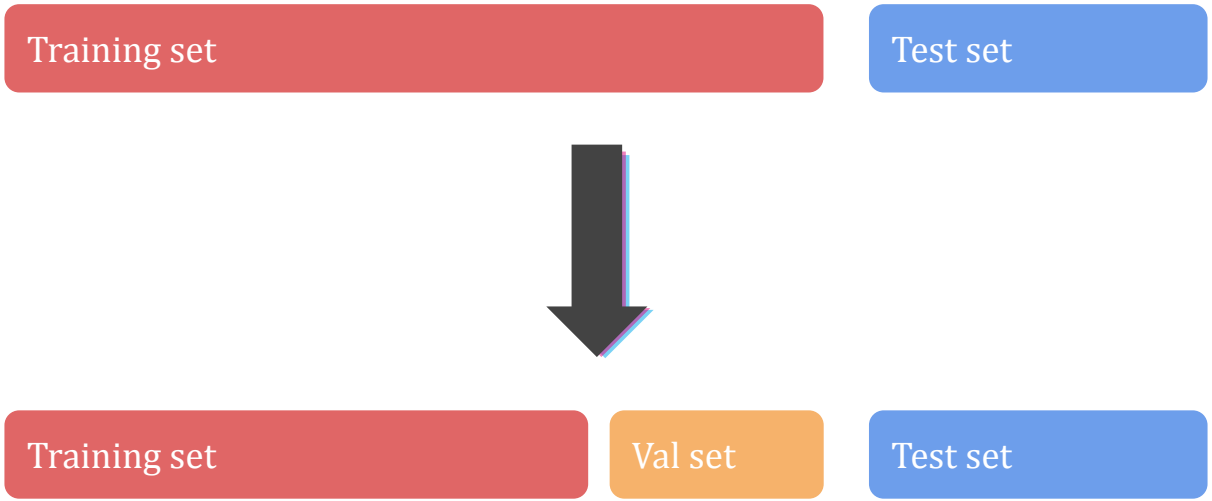
Normalmente, nós realizamos dezenas ou centenas de experimentos com modelos diferentes e escolhemos qual funciona melhor no conjunto de Validação.

Um problema com isso, é que testando tantas variações, podemos encontrar algum modelo que funciona muito bem na validação mas não tão bem em dados novos.

Treino, Validação e Teste

Para combater isso, nós separamos um terceiro conjunto chamado de Teste. O conjunto de Teste será usado para medir a performance **final** do nosso modelo, depois de terminarmos todos os experimentos com a validação.

Validation Set (conjunto de validação)



Evitando o overfitting no teste

É importante que usemos o conjunto de teste **o mínimo** possível, para evitar o mesmo efeito dos nossos modelos se adaptarem ao teste. Além disso, depois de avaliar um modelo no teste **não podemos** fazer alterações nos hiperparâmetros e avaliarmos de novo.

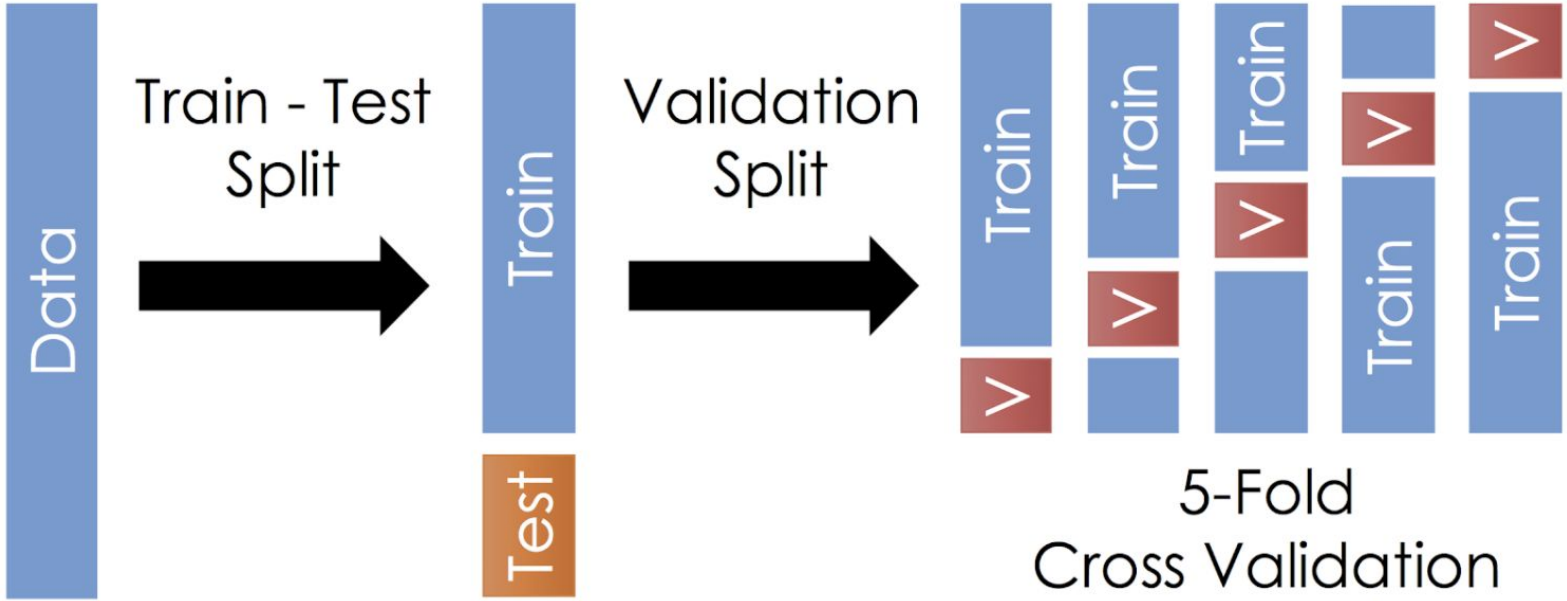
Problemas com essa separação

- Se o conjunto de validação for muito pequeno, os valores dos resultados não vão ser tão confiáveis
- Se o conjunto de validação for muito grande, teremos menos dados para treinar o modelo

Cross-Validation

Para combater isso, podemos dividir o dataset de treino em k partes. Então nós treinamos o modelo k vezes, cada vez usando uma parte diferente como validação, usando a média final como a métrica do modelo.

Cross Validation



Vantagens e Desvantagens

Vantagens:

- As métricas calculadas são muito mais confiáveis e têm menos ruído, já que estamos usando a média de k experimentos diferentes
- Quando decidirmos os hiperparâmetros finais, podemos treinar um modelo em todas as k partes e avaliarmos no teste

Desvantagens:

- Precisamos treinar o mesmo modelo k vezes para cada experimento! Isso pode ficar inviável para modelos mais pesados.

Hiperparâmetros

A maioria dos modelos de aprendizado de máquina tem diversos hiperparâmetros que podem afetar o seu funcionamento. Para encontrar a melhor combinação, podemos seguir o seguinte procedimento:

1. Treinamos diversos modelos com diferentes combinações de hiperparâmetros
2. Avaliamos os modelos treinados no conjunto de validação
3. Ao final de todos os experimentos, avaliamos o modelo selecionado no conjunto de teste

Busca por hiperparâmetros

Para modelos com muitos hiperparâmetros, não é possível testar todas as combinações possíveis para avaliar qual funciona melhor.

Em vez disso, precisamos de estratégias para encontrar boas configurações sem necessitar de uma quantidade absurda de experimentos.

Grid Search

A Grid Search (busca em grade) cria uma grade, com cada ponto representando uma combinação de diferentes hiperparâmetros, e avalia o modelo para cada ponto.

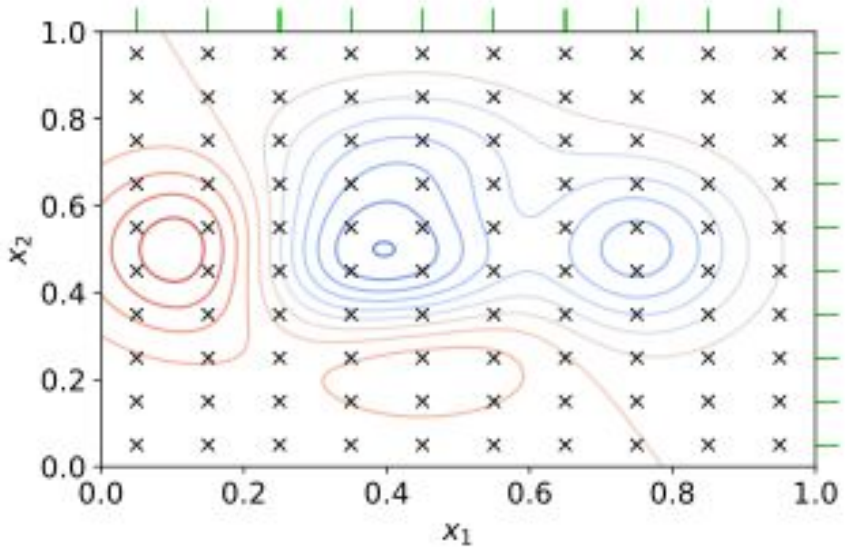
Ela tem a vantagem de fazer uma exploração extensa do espaço de busca, mas pode ser extremamente custosa para modelos com muitos hiperparâmetros.

Grid Search

Na Grid Search, existe um tradeoff entre custo computacional e o quanto o espaço de busca é explorado:

- Se diminuirmos o número de possibilidades para cada hiperparâmetro, a grade terá menos pontos a serem considerados, mas não estaremos explorando tão bem os espaços entre cada ponto
- Se aumentarmos o número de possibilidades, exploraremos mais o espaço, mas também teremos muito mais pontos a considerar

Grid Search



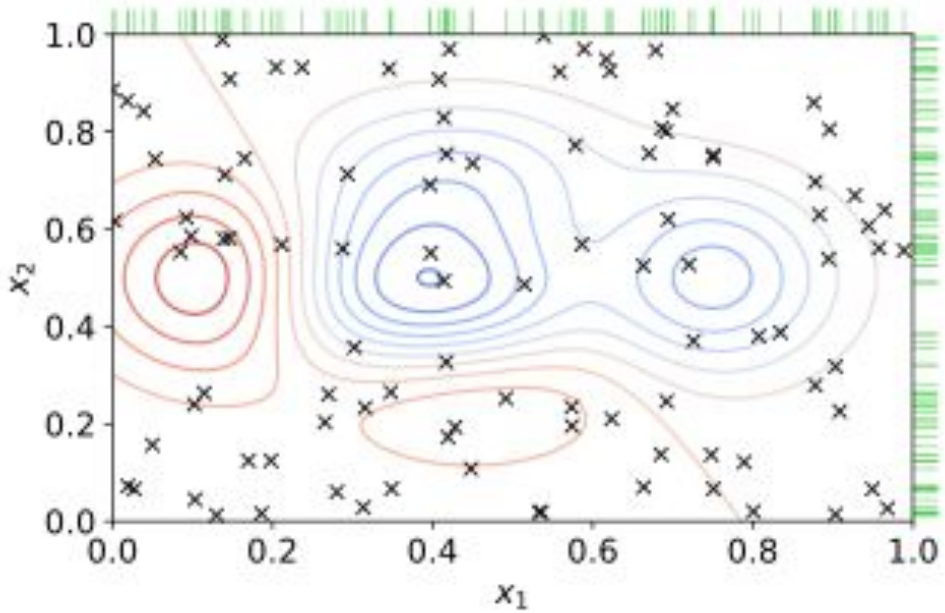
Grid Search para dois hiperparâmetros, considerando 10 valores diferentes para cada.

Busca Aleatória

Para modelos com muitos hiperparêmetros e com alto custo computacional (como redes neurais), é difícil usar a busca em grade pelo seu alto custo computacional.

Em vez disso, podemos usar a busca aleatória: encontramos N pontos aleatórios no espaço de busca e avaliamos o modelo para esses valores.

A busca aleatória pode conseguir resultados bons muito mais rapidamente, mas pode não explorar tanto o espaço de busca.



A busca aleatória pode obter resultados muito mais rapidamente, mas não tem as mesmas garantias da busca em grade.

Problemas com os métodos de busca

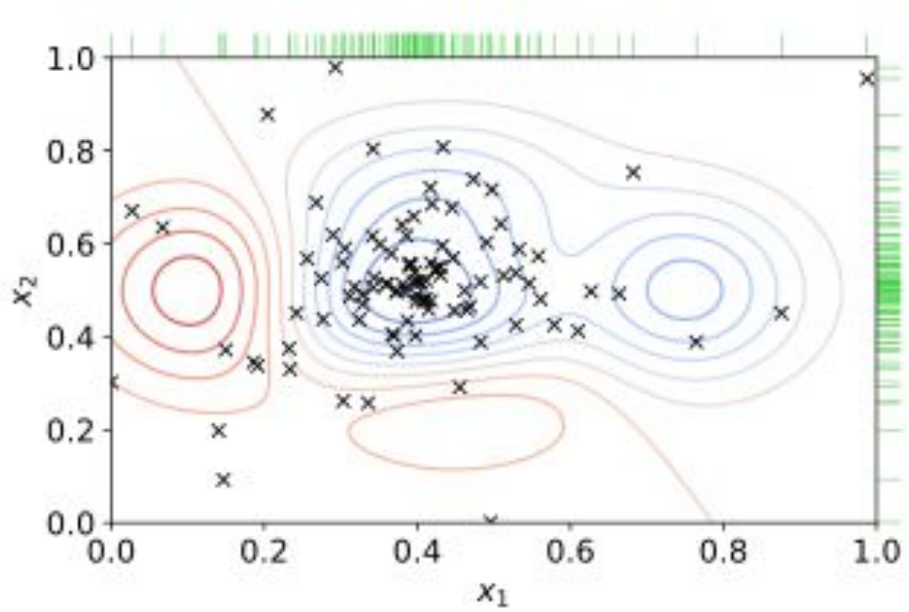
Um método ideal de busca teria algumas propriedades que não estão presentes nem na busca em grade ou na busca aleatória:

- Nós idealmente queremos explorar mais as regiões que são mais promissoras ou as que nós não ainda temos informações
- Nós não queremos gastar poder computacional com regiões que sabemos que irão ter resultados ruins

Métodos inteligentes de busca

Para corrigir esses problemas, existem métodos mais sofisticados para realizar a busca por hiperparâmetros:

- Na Otimização Bayesiana, um modelo probabilístico encontra configurações que revelam o máximo de informação possível a respeito do espaço de busca
- Algoritmos Evolutivos exploram preferencialmente as regiões que se mostraram mais efetivas nas populações anteriores



Em métodos como a Otimização Bayesiana, podemos explorar o espaço de busca de maneira inteligente, considerando as observações anteriores.