QUAL O LIMITE DA DIMENSIONALIDADE?

Que bom que você está aqui. Espero que tenha conseguido treinar um pouco de tudo o que vimos e compreender os conceitos por trás das técnicas e algoritmos que vimos até aqui. A construção do arsenal de qualquer Data Scientists passa por uma primeira etapa de familiaridade e ganha uma qualidade exponencial com o uso, testes, ajustes. Ou seja, tentem experimentar um pouco de tudo o que já vimos. Esse é o momento de aprender. Compartilhe as dificuldades e aprendizados no grupo do slack e você vai ver a mágica do conhecimento e aprendizado coletivo em ação. Nos últimos textos falamos bastante sobre como tornar o modelo mais estável e complexo, seja pela engenharia de features ou pelo agrupamento de algoritmos. Estamos falando mais especificamente tanto de Feature Engineering quanto de Ensembles.

Neste texto vamos falar de Redução de dimensionalidade, algo que pode parecer contraditório em relação a tudo o que vimos em Feature Engineering. Mas essas técnicas de aprendizado não supervisionado são fundamentais na caixa de ferramentas de profissionais de ciência de dados.

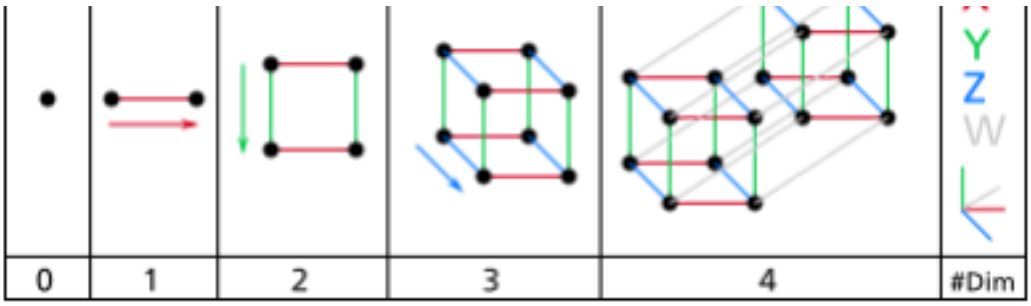
Para facilitar a compreensão das explicações, baixe os datasets utilizados na elaboração deste texto e um notebook com os códigos **aqui**.

A MALDIÇÃO DA DIMENSIONALIDADE

Já falamos um pouco sobre o conceito de dimensionalidade e redução de dimensionalidade no texto Learning>Machine, que abordava os temas de aprendizado supervisionado e não supervisionado. Nele, vimos que cada nova feature ou coluna representa uma dimensão. Isso pode causar problemas nos mais diversos aspectos como:

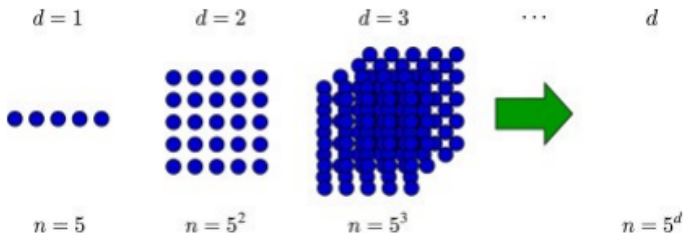
- A dificuldade de visualizar objetos com mais dimensões do que 3:

QUAL O LIMITE DA DIMENSIONALIDADE?



Complexidade ao interpretar objetos com mais de 4 dimensões.
Fonte:https://en.wikipedia.org/wiki/File:Dimension_levels.svg

- O aumento da complexidade e necessidade computacional, tanto para processar, quanto para armazenar esses dados.



Representação do aumento exponencial de cada nova feature adicionada a um dataset com 5 observações.
Fonte: Masashi Sugiyama, in Introduction to Statistical Machine Learning, 2016

AVANÇAR

PRINCIPAIS PROBLEMAS CAUSADOS NOS ALGORITMOS

ALTA COMPLEXIDADE

Imagine que o seu dataset possui em uma feature mais de 50 categorias diferentes. Quando isto acontece, dizemos que esta coluna possui alta cardinalidade. Se decidirmos mantê-la, ao passarmos pelo one-hot encoding, automaticamente nosso dataset passou a ter mais 50 (ou 49 se usarmos o conceito de dummies) dimensões. Sabendo que estamos falando do acréscimo do número de observações elevado à 50ª potência, intuitivamente já percebemos que não é uma boa ideia treinar um modelo e fazer previsões com um dataset tão complexo quanto este.

ESPARSIDADE DOS DADOS

Um outro problema comum é que os dados gerados são muito esparsos. Imagine que em 50 colunas, teremos 49 zeros e apenas 1 número um. Isto é difícil para diversos algoritmos calcular e estabelecer uma significância estatística. Em outras palavras, seria necessário um conjunto muito grande de observações o que tornaria ainda mais complexo o dataset.

DECORAR X APRENDER

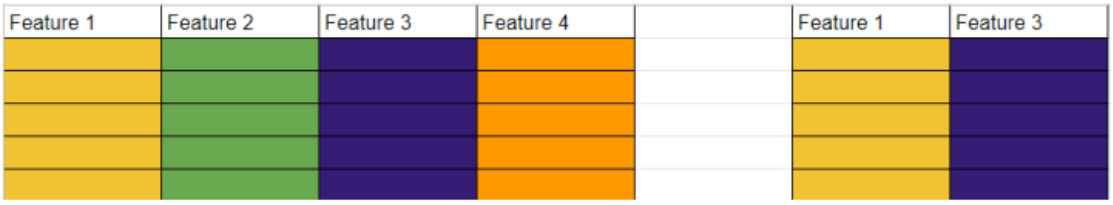
Vamos supor que tomamos a decisão de usar um dataset com um número elevado de dimensões. Existe uma grande probabilidade que pouquíssimas observações tenham a mesma combinação de features. Por exemplo em um dataset de carros, onde precisaríamos prever o seu valor de venda baseado em suas features, se tivéssemos 30 features como: cor, marca, modelo, ano, com_adesivo, com_roda, com_defeito_maçaneta, mais_50km… provavelmente chegaríamos à combinações em que teríamos no máximo 2 ou 3 observações. E isso, ao invés de ajudar os algoritmos a aprender para generalizar posteriormente, os leva a um overfitting.

ESTRATÉGIAS PARA RESOLVER A ALTA DIMENSIONALIDADE

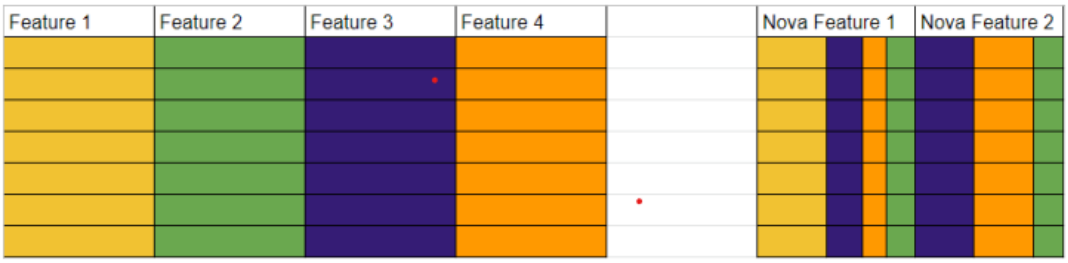
Agora que já falamos dos perigos da alta dimensionalidade, vamos entender as estratégias e técnicas que podemos usar para tratar, combater e/ou

QUAL O LIMITE DA DIMENSIONALIDADE?

fazem a diferença e ajudam o algoritmo a aprender, descartando as demais. Na segunda, gerar novas features que são uma derivação das iniciais, capturam informações delas, mas não possuem mais as mesmas interpretações. Veja a imagem abaixo para ter um pouco da intuição por trás desses dois conceitos.



Exemplo de Feature Selection onde escolhemos algumas features em detrimento à outras. Pode haver uma perda sensível de informações.



Exemplo de Feature Extraction, onde as 4 features originais são usadas para derivar duas novas features.

FEATURE SELECTION

Existem algumas técnicas que podemos usar para realizar uma feature selection. Em geral, é importante que haja um conhecimento sobre o dataset e o problema de negócio que você está atacando. Caso contrário, pode ser que exclua alguma feature importante que vai fazer falta no modelo em produção, por exemplo.

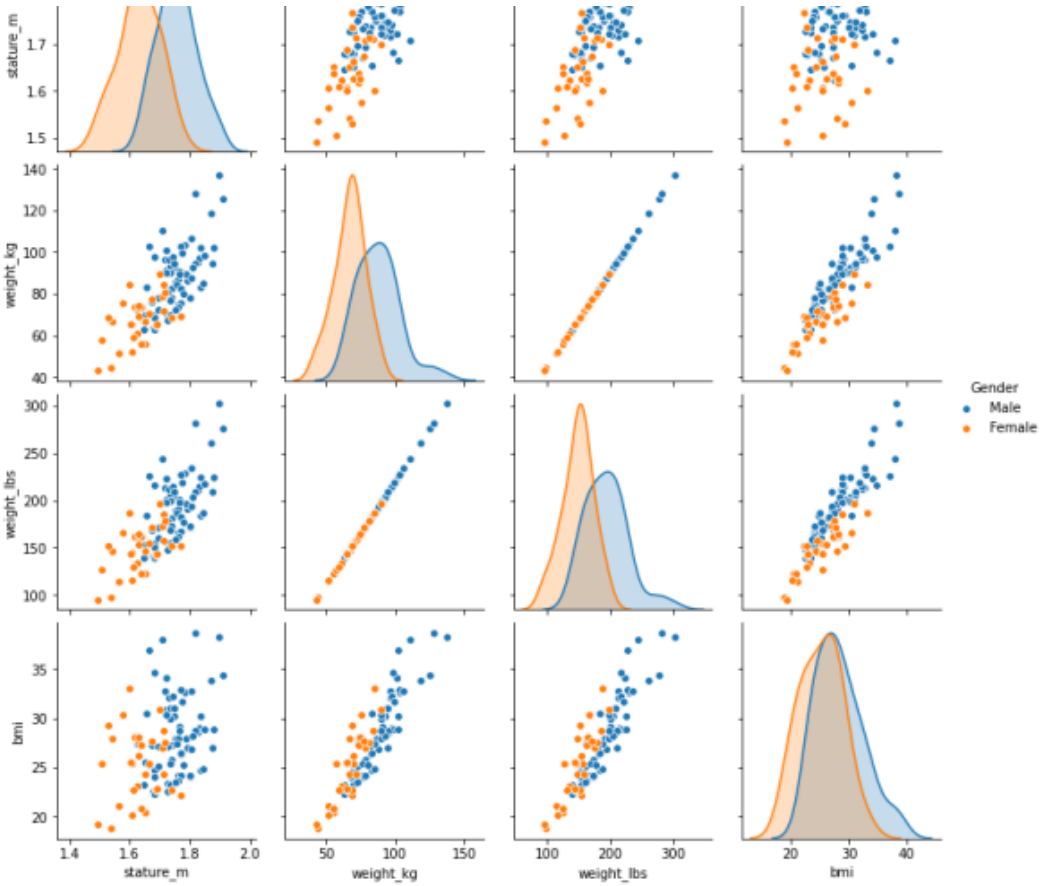
AVANÇAR

SELEÇÕES

SELEÇÃO POR CORRELAÇÃO

Ao fazer uma análise exploratória podemos identificar visualmente duas ou mais variáveis que são absolutamente correlacionadas. Muitas vezes pode ser uma transformação de dados ou uma outra forma de derivada de capturar a mesma informação. O fato é que ao variarem de uma maneira igual, elas registram exatamente a mesma informação ao dataset. Para exemplificar, pegamos o dataset Ansur com informações de homens e mulheres das forças armadas americanas. Vamos pegar um subset do dataset com apenas 3 features originais (altura, peso em kg e gênero) e 2 features criadas a partir delas (peso em libras e o IMC). Vamos enxergar como essas correlações ficam visivelmente claras:

QUAL O LIMITE DA DIMENSIONALIDADE?



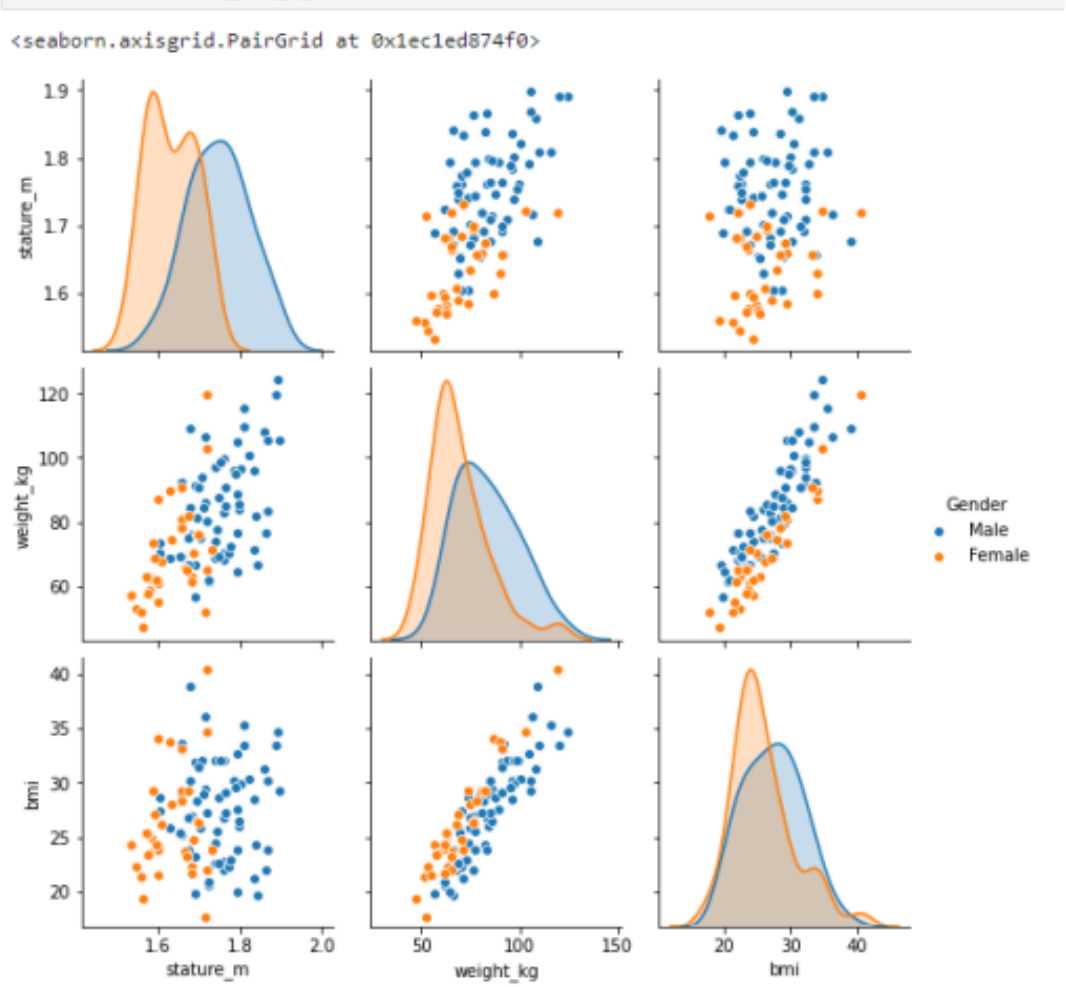
Perceba que temos uma relação linear perfeita entre o peso em libras e o peso em kilogramas.

```
ansur_bmi.corr()
```

	stature_m	weight_kg	weight_lbs	bmi
stature_m	1.000000	0.660265	0.660265	0.183094
weight_kg	0.660265	1.000000	1.000000	0.854700
weight_lbs	0.660265	1.000000	1.000000	0.854700
bmi	0.183094	0.854700	0.854700	1.000000

Precisamos excluir uma delas (nesse caso, vamos fazer um drop na coluna do peso em libras e manter o peso em kg, por uma questão de familiaridade nossa com a medida):

QUAL O LIMITE DA DIMENSIONALIDADE?



Ainda há uma correlação perfeita entre as features, mas ela não é linear. Essa relação obedece a fórmula de cálculo do IMC : peso / altura²

Dependendo da análise que se vai fazer, é possível manter apenas o IMC e descartar as duas features de peso e altura, já que estas últimas foram capturadas quase em sua totalidade pelo IMC. Mas existem outras formas mais computacionais e estatísticas de fazer uma feature selection.

SELEÇÃO DE FEATURES POR VARIÂNCIA

Como já vimos, a variância é um dos elementos que mais impacta na qualidade e performance dos modelos. Logo, features com maiores variâncias vão impactar mais e ser mais relevantes para o nosso modelo do que features com menos variância. Dessa vez, vamos usar todas as features numéricas do dataset Ansur.

QUAL O LIMITE DA DIMENSIONALIDADE?

```
Index(['Branch', 'Component', 'Gender', 'abdominalextensiondepthsitting',
       'acromialheight', 'acromionradialelength', 'anklecircumference',
       'axillaheight', 'balloffootcircumference', 'balloffootlength',
       'biacromialbreadth', 'bicepscircumferenceflexed', 'bicristalbreadth',
       'bideltoidbreadth', 'bimalleolarbreadth', 'bitragionchinarc',
       'bitragionsubmandibulararc', 'bizygomaticbreadth',
       'buttockcircumference', 'buttockdepth', 'buttockheight',
       'buttockkneelength', 'buttockpoplitealength', 'calfcircumference',
       'cervicaleheight', 'chestbreadth', 'chestcircumference', 'chestdepth',
       'chestheight', 'crotchheight', 'crotchlengthomphalion',
       'crotchlengthposterioromphalion', 'earbreadth', 'earlength',
       'earprotrusion', 'elbowrestheight', 'eyeheightsitting',
       'footbreadthhorizontal', 'footlength', 'forearmcenterofgriplength',
       'forearmcircumferenceflexed', 'forearmforearmbreadth',
       'forearmhandlength', 'functionalleglength', 'handbreadth',
       'handcircumference', 'handlength', 'headbreadth', 'headcircumference',
       'headlength', 'heelanklecircumference', 'heelbreadth', 'hipbreadth',
       'hipbreadthsitting', 'iliocristaleheight', 'interpupillarybreadth',
       'interscyei', 'interscyeii', 'kneeheightmidpatella',
       'kneeheightsitting', 'lateralfemoralepicondyleheight',
       'lateralmalleolusheight', 'lowerthighcircumference',
       'mentonsellionlength', 'neckcircumference', 'neckcircumferencebase',
       'overheadfingertipreachsitting', 'palmlength', 'poplitealheight',
       'radialestylionlength', 'shouldercircumference', 'shoulderelbowlength',
       'shoulderlength', 'sittingheight', 'sleevelengthspinewrist',
       'sleeveoutseam', 'span', 'suprasternaleheight', 'tenthribheight',
       'thighcircumference', 'thighclearance', 'thumbtipreach', 'tibialheight',
       'tragiontopofhead', 'trochanterionheight',
       'verticaltrunkcircumferenceusa', 'waistbacklength', 'waistbreadth',
       'waistcircumference', 'waistdepth', 'waistfrontlengthsitting',
       'waistheightomphalion', 'wristcircumference', 'wristheight',
       'weight_kg', 'stature_m', 'BMI', 'BMI_class', 'Height_class'],
      dtype='object')
```

O dataset Ansur completo possui 99 colunas, sendo 94 numéricas e 5 categóricas.

Um ponto importante para que essa seleção por threshold tenha efeito é que todos os dados estejam normalizados. Isso porque precisamos capturar apenas a variação dentro de cada coluna as relações entre das diferentes escalas. Por exemplo, o tamanho da mão é muito menor em centímetros que o tamanho da tíbia, mas ao normalizar, preservamos apenas a variância dentro de cada coluna. Para normalizar as features, basta dividirmos o valor da observação pela média da coluna em questão. Vamos aos cálculos:

- **Importando VarianceThreshold**

  from sklearn.feature_selection import VarianceThreshold

- **Criando VarianceThreshold feature selector com 0.005**

  sel = VarianceThreshold(threshold=0.005)

- **Fitando o selector ao df numeric normalizado**

  sel.fit(df_numeric / df_numeric.mean())

QUAL O LIMITE DA DIMENSIONALIDADE?

mask = sel.get_support()

- **Apply the mask to create a reduced dataframe**

reduced_ansur = df_numeric.loc[:, mask]

```
reduced_ansur.shape
```

```
(6068, 45)
```

Wow! Com um threshold de apenas 0.005 de variância nosso dataset foi de 94 features para 45. É uma redução e tanto. Esse método é uma espécie de força bruta e só deve ser usado quando se conhece muito bem o dataset e os problemas de negócio que se está resolvendo. Mas podemos melhorar ainda mais a forma de seleção.

SELEÇÃO DE FEATURES POR PERFORMANCE DO MODELO

Uma das formas mais intuitivas de escolher as features em machine learning é através do impacto delas no modelo que estamos usado. Para este exemplo vamos usar o dataset Pima Indians editado. Nele temos 8 features com as pré-condições médicas de 392 pacientes e o objetivo é prever na coluna test se o paciente terá ou não diabetes. Para que possamos prever de forma precisa, vamos usar o StandardScaler() que vai padronizar os dados centrando a média de todas as features em 0 e a variância máxima em 1. Vamos fazer as importações necessárias e criar os datasets de treino e test.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
pima = pd.read_csv('Pimaindians.csv')
```

```
pima.columns
```

```
Index(['pregnant', 'glucose', 'diastolic', 'triceps', 'insulin', 'bmi',
       'family', 'age', 'test'],
      dtype='object')
```

```
pima.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 392 entries, 0 to 391
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pregnant   392 non-null    int64
 1   glucose    392 non-null    int64
 2   diastolic  392 non-null    int64
 3   triceps    392 non-null    int64
 4   insulin    392 non-null    int64
 5   bmi        392 non-null    float64
 6   family     392 non-null    float64
 7   age        392 non-null    int64
 8   test       392 non-null    object
dtypes: float64(2), int64(6), object(1)
memory usage: 27.7+ KB
```

```
X = pima.drop(columns=['test'])
y = pima['test']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=0)
```

- **Instanciando o StandardScaler e a Regressão logística**

scaler = StandardScaler()

lr = LogisticRegression()

QUAL O LIMITE DA DIMENSIONALIDADE?

- **Treinando a regressão linear no X_train padronizado**

  lr.fit(X_train_std, y_train)

- **Precisamos transformar os dados de test também** X_test_std = scaler.transform(X_test)

- **Fazendo predições com os dados padronizados**

  y_pred = lr.predict(X_test_std)

- **imprimindo as métricas de acurácia e os coeficientes de importância das features**

  print("{0:.1%} accuracy on test set.".format(accuracy_score(y_test, y_pred)))

  print(dict(zip(X.columns, abs(lr.coef_[0]).round(2))))

AVANÇAR