

Relatório <010> - < 10 - Prática: Lidando com Dados do Mundo Real (II) >

<Kawan Machado>

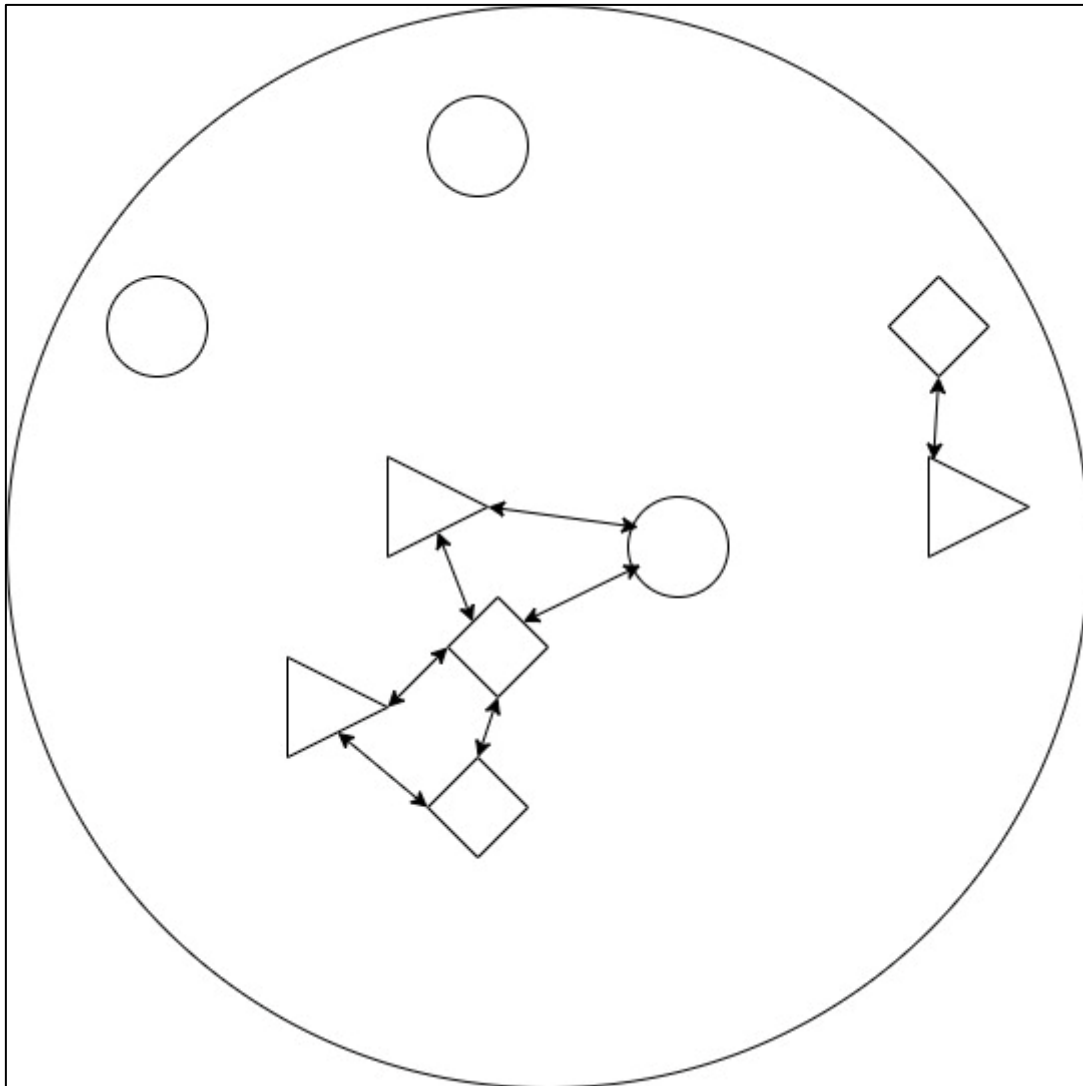
Descrição da atividade

1. More Data Mining and Machine Learning Techniques

1.1 K-Nearest-Neighbors Concepts

Iniciando por esse conceito, fica evidente como as técnicas de aprendizado de máquina podem ser tão simples. A ideia é classificar os pontos dos dados baseado na distância entre si. Para cada nova entrada, o algoritmo classifica a distância dos vizinhos mais próximos e agrupa os baseados em suas distâncias.

FIGURA 1 - K-NEAREST-NEIGHBORS



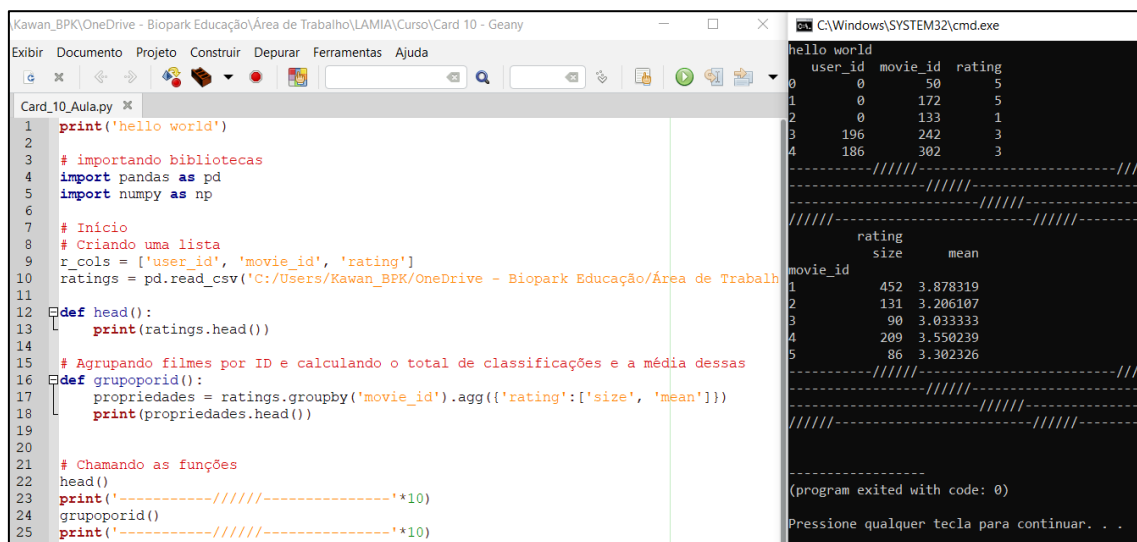
Elaborado pelo próprio autor, 2025.

O KNN é um dos modelos mais simples de aprendizado de máquina, ele é classificado como aprendizado supervisionado. No próximo tópico vamos explorar a classificação de filmes baseado nos seus metadados.

1.2 Activity Using KNN to predict a rating for a movie

Para esse exemplo vamos tratar o data set 'MovieLens', com ele vamos classificar os filmes olhando os 10 mais próximos em termos de gênero e popularidade e com isso vamos conseguir prever a avaliação média esperada de um filme do mesmo gênero. Primeiro fazemos a importação dos dados e depois calculamos a quantidade e a média de classificação dos filmes.

FIGURA 2 – IMPORTAÇÃO E PRIMEIRO CÁLCULO



```
1 print('hello world')
2
3 # importando bibliotecas
4 import pandas as pd
5 import numpy as np
6
7 # Inicio
8 # Criando uma lista
9 r_cols = ['user_id', 'movie_id', 'rating']
10 ratings = pd.read_csv('C:/Users/Kawan_BPK/OneDrive - Biopark Educação/Área de Trabalho/LAMIA/Curso/Card 10 - Geany
11
12 def head():
13     print(ratings.head())
14
15 # Agrupando filmes por ID e calculando o total de classificações e a média dessas
16 def grupopord():
17     propriedades = ratings.groupby('movie_id').agg({'rating': ['size', 'mean']})
18     print(propiedades.head())
19
20
21 # Chamando as funções
22 head()
23 print('-----//----*10)
24 grupopord()
25 print('-----//----*10)
```

```
hello world
user_id  movie_id  rating
0        0         50      5
1        0        172      5
2        0        133      1
3       196        242      3
4       186        302      3
-----//----*10)
rating
size    mean
movie_id
1        452  3.878319
2        131  3.206107
3         90  3.033333
4        209  3.550239
5         86  3.302326
-----//----*10)
(program exited with code: 0)
Pressione qualquer tecla para continuar. . .
```

Elaborado pelo próprio autor, 2025.

A média bruta de avaliações dos filmes pode conter problemas, pois alguns filmes tem mais avaliações que outros. Para isso efetuamos a normalização dos valores para eles ficarem entre 0 e 1, com isso conseguimos prosseguir a análise.

FIGURA 3 – NORMALIZAÇÃO DOS DADOS

```

1 print('hello world')
2
3 # importando bibliotecas
4 import pandas as pd
5 import numpy as np
6
7 # Inicio
8 # Criando uma lista
9 r_cols = ['user_id', 'movie_id', 'rating']
10 ratings = pd.read_csv('C:/Users/Kawan_BPK/OneDrive - Biopark Educação/Área de Trabalho/LAMIA/Curso/Card 10 - Geany
11
12 def head():
13     print(ratings.head())
14
15 # Agrupando filmes por ID e calculando o total de classificações e a média dessas
16 def grupoporid():
17     propriedades = ratings.groupby('movie_id').agg({'rating': ['size', 'mean']})
18     return propriedades
19
20 # O número bruto de classificações não é muito útil para calcular as distancias entre
21
22 def normalizados():
23     propriedades = grupoporid()
24     filmesqtdmedia = pd.DataFrame(propiedades['rating']['size'])
25     filmesnormalqtdmedia = filmesqtdmedia.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
26     return filmesnormalqtdmedia
27
28 def generosfilme():
29     movieProperties = ratings.groupby('movie_id').agg({'rating': ['size', 'mean']})
30     movieNumRatings = pd.DataFrame(movieProperties['rating']['size'])
31     movieNormalizedNumRatings = movieNumRatings.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
32     movieDict = {}
33     with open('C:/Users/Kawan_BPK/OneDrive - Biopark Educação/Área de Trabalho/LAMIA/Curso/Card 10 - Geany
34         temp = ''
35         for line in f:
36             fields = line.rstrip('\n').split('|')
37             movieID = int(fields[0])
38             name = fields[1]
39             genres = fields[5:25]
40             genres = list(map(int, genres))
41             movieDict[movieID] = (name, genres, movieNormalizedNumRatings.loc[movieID])
42     return movieDict
43

```

```

movie_id  size  mean
452      3.878319
131      3.286107
90       3.833333
209      3.550239
86       3.382326

movie_id  size
0       0.773585
1       0.222985
2       0.152659
3       0.356775
4       0.145798

```

Elaborado pelo próprio autor, 2025.

Agora precisamos importar a informação dos gêneros dos filmes que estão contidos no arquivo “u.file” e mesclar com os valores das médias de avaliação e com os valores normalizados das mesmas. Para exemplificar trouxemos os valores do item 1, que corresponde ao filme ‘Toy Story’ que se encaixa em três genêros, tem uma média de avaliações normalizada de 0,77 e uma média bruta de 3,87.

FIGURA 4 – DADOS DO FILME 1

```

15
16 # Agrupando filmes por ID e calculando o total de classificações e a média dessas
17 def grupoporid():
18     propriedades = ratings.groupby('movie_id').agg({'rating': ['size', 'mean']})
19     return propriedades
20
21 # O número bruto de classificações não é muito útil para calcular as distancias entre
22
23 def normalizados():
24     propriedades = grupoporid()
25     filmesqtdmedia = pd.DataFrame(propiedades['rating']['size'])
26     filmesnormalqtdmedia = filmesqtdmedia.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
27     return filmesnormalqtdmedia
28
29 def generosfilme():
30     movieProperties = ratings.groupby('movie_id').agg({'rating': ['size', 'mean']})
31     movieNumRatings = pd.DataFrame(movieProperties['rating']['size'])
32     movieNormalizedNumRatings = movieNumRatings.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
33     movieDict = {}
34     with open('C:/Users/Kawan_BPK/OneDrive - Biopark Educação/Área de Trabalho/LAMIA/Curso/Card 10 - Geany
35         temp = ''
36         for line in f:
37             fields = line.rstrip('\n').split('|')
38             movieID = int(fields[0])
39             name = fields[1]
40             genres = fields[5:25]
41             genres = list(map(int, genres))
42             movieDict[movieID] = (name, genres, movieNormalizedNumRatings.loc[movieID])
43     return movieDict
44

```

```

movie_id  size
0       0.773585
1       0.222985
2       0.152659
3       0.356775
4       0.145798

('Toy Story (1995)', [0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], np.float64(0.7735849056037774), np.float64(3.8783185840707963))

```

Elaborado pelo próprio autor, 2025.

Agora vamos computar a distância entre dois filmes, no caso o filme 2 e o filme 4. O resultado é um valor de 0,80, lembrando que quanto maior menos similar um filme é de outro, nesse caso, 0,80 significa que os filmes não são parecidos.

FIGURA 5 – DISTÂNCIA ENTRE FILME 2 E 4

```

26 filmesqtmedia = pd.DataFrame(propriedades['rating']['size'])
27 filmesnormalqtmedia = filmesqtmedia.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
28 return filmesnormalqtmedia
29 # parte 1 - filme1
30 def generosfilme():
31     movieProperties = ratings.groupby('movie_id').agg({'rating': ['size', 'mean']})
32     movieNumRatings = pd.DataFrame(movieProperties['rating']['size'])
33     movieNormalizedNumRatings = movieNumRatings.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
34     movieDict = {}
35     with open('C:/Users/Kawan_BKF/OneDrive - Biopark Educação/Área de Trabalho/LAMIA/Curso/Card 10 - Geany') as f:
36         temp = ''
37         for line in f:
38             fields = line.rstrip('\n').split('|')
39             movieID = int(fields[0])
40             name = fields[1]
41             genres = fields[5:25]
42             genres = list(map(int, genres))
43             movieDict[movieID] = (name, genres, movieNormalizedNumRatings.loc[movieID].get('mean'))
44     return movieDict[1]
45 # parte 2 - distancia entre dois filmes
46 movieProperties = ratings.groupby('movie_id').agg({'rating': ['size', 'mean']})
47 movieNumRatings = pd.DataFrame(movieProperties['rating']['size'])
48 movieNormalizedNumRatings = movieNumRatings.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
49 movieDict = {}
50 with open('C:/Users/Kawan_BKF/OneDrive - Biopark Educação/Área de Trabalho/LAMIA/Curso/Card 10 - Geany') as f:
51     temp = ''
52     for line in f:
53         fields = line.rstrip('\n').split('|')
54         movieID = int(fields[0])
55         name = fields[1]
56         genres = fields[5:25]
57         genres = list(map(int, genres))
58         movieDict[movieID] = (name, genres, movieNormalizedNumRatings.loc[movieID].get('mean'))

```

```

movie_id size
1 0.773585
2 0.222985
3 0.152659
4 0.356775
5 0.145798

('Toy Story (1995)', [0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], np.float64(0.7735849056603774), np.float64(3.8783185840707963))
0.8004574042309892

Liar Liar (1997) 3.156701030927835
Aladdin (1992) 3.8127853881278537
Willy Wonka & the Chocolate Factory (1971) 3.6319018404907975
Monty Python and the Holy Grail (1974) 4.0664556962025316
Full Monty, The (1997) 3.926984126984127

```

Elaborado pelo próprio autor, 2025.

Por fim, vamos usar o filme 1 ('toy sotry') para agrupar todos os filmes do mesmo gênero e calcular o valor médio de avaliação destes e com isso predizer a média de avaliação esperada para um filme do mesmo grupo.

FIGURA 6 – MÉDIA DE AVALIAÇÃO

```

56 genres = fields[5:25]
57 genres = list(map(int, genres))
58 movieDict[movieID] = (name, genres, movieNormalizedNumRatings.loc[movieID].get('mean'))
59 # parte 3 distancia entre filmes em relação ao toy story
60 def ComputeDistance(a,b):
61     genresA = a[1]
62     genresB = b[1]
63     genresDistance = spatial.distance.cosine(genresA, genresB)
64     popularityA = a[2]
65     popularityB = b[2]
66     popularityDistance = abs(popularityA - popularityB)
67     return genresDistance + popularityDistance
68
69 def getNeighbors(movieID, K):
70     distances = []
71     for movie in movieDict:
72         if movie != movieID:
73             dist = ComputeDistance(movieDict[movieID], movieDict[movie])
74             distances.append((movie, dist))
75     distances.sort(key=operator.itemgetter(1))
76     neighbors = []
77     for x in range(K):
78         neighbors.append(distances[x][0])
79     return neighbors
80
81 def abc():
82     K = 10
83     avgRating = 0
84     neighbors = getNeighbors(1, K)
85     for neighbor in neighbors:
86         avgRating += movieDict[neighbor][3]
87     print(movieDict[neighbor][0] + " " + str(movieDict[neighbor][3]))
88     avgRating /= float(K)
89     print("-----" * 10)
90     return avgRating
91
92 # Chamando a função
93 abc()

```

```

movie_id size
1 0.773585
2 0.222985
3 0.152659
4 0.356775
5 0.145798

('Toy Story (1995)', [0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], np.float64(0.7735849056603774), np.float64(3.8783185840707963))
0.8004574042309892

Liar Liar (1997) 3.156701030927835
Aladdin (1992) 3.8127853881278537
Willy Wonka & the Chocolate Factory (1971) 3.6319018404907975
Monty Python and the Holy Grail (1974) 4.0664556962025316
Full Monty, The (1997) 3.926984126984127
George of the Jungle (1997) 2.685185185185185
Beavis and Butt-head Do America (1996) 2.7884615384615383
Birdcage, The (1996) 3.44168608629385
Home Alone (1990) 3.0875912408759123
Aladdin and the King of Thieves (1996) 2.8461538461538463
3.3445905900235564

```

Elaborado pelo próprio autor, 2025.

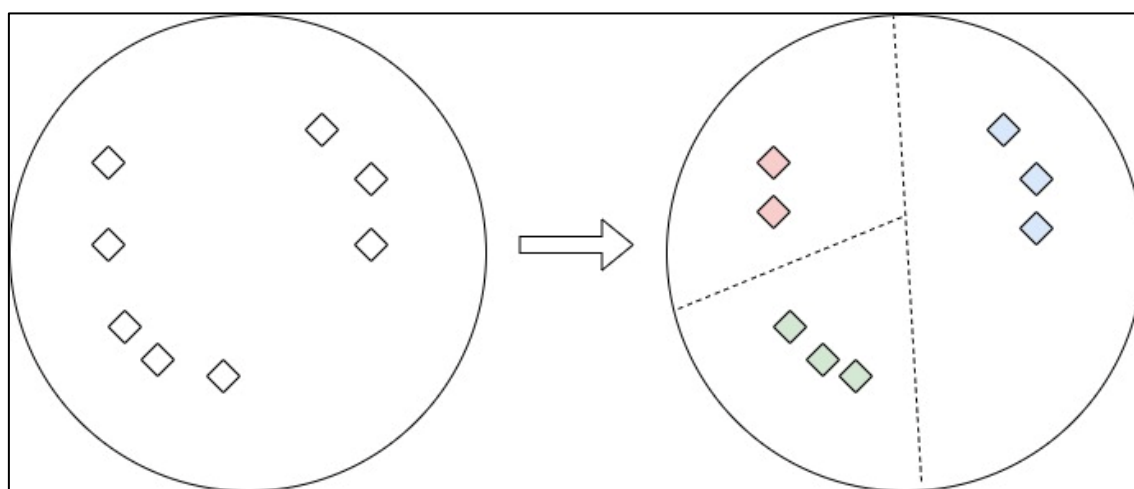
Observando a imagem acima, primeiro temos os filmes que se encaixaram no mesmo grupo do filme 'toy story' e abaixo temos a média de avaliações '3,34', com isso podemos dizer que a avaliação média esperada para um filme que se encaixa no mesmo padrão de gênero de 'toy story' é de 3,34.

1.3 Dimensionality Reduction; Principal Component Analysis (PCA)

Explorando os temas de dimensionalidades, o primeiro item que chama a atenção é a maldição da dimensionalidade, ou seja, um cálculo matemático que pode gerar infinitas dimensões, por exemplo, em um sistema de recomendações, o vetor de avaliações para cada filme pode representar uma dimensão, ou seja, cada filme possuiria sua própria dimensão, o que pode comprometer recursos computacionais.

Para tratar dessa situação, é feita a redução da dimensionalidade, um dado em grandes dimensões reduzido a uma dimensão menor preservando o máximo de variância que for possível. Uma forma de fazer isso é usando K-Means, se cada filme é a sua própria dimensão, agrupar eles reduzem a dimensão individual para uma dimensão de grupos.

FIGURA 7 – K-MEANS REDUÇÃO DE DIMENSIONALIDADES



Elaborado pelo próprio autor, 2025.

Outra forma de diminuir a dimensionalidade é através da Análise de Componentes Principais (PCA), ela envolve matemática complexa, mas em termos simples, ela identifica autovetores nos dados em alta dimensão, os quais definem hiperplanos que dividem os dados, preservando ao máximo a variância (diferença) entre eles. Os dados então são projetados nesses hiperplanos que representam dimensões reduzidas.

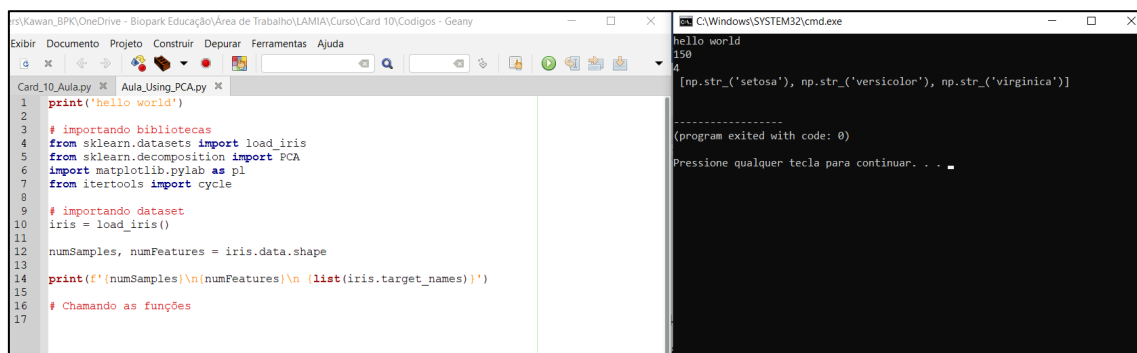
Uma implementação dessa técnica é o algoritmo, Decomposição em Valores Singulares (SVD). Essa técnica é usada para a compressão de imagens e reconhecimento facial. Um exemplo é a visualização de dados do conjunto íris, ela contém dados das pétalas e sépalas de diversas flores de íris.

Nesses dados temos o comprimento e a largura das pétalas e o comprimento e a largura das sépalas. Isso significam 4 dimensões, se quisermos plotar isso em um gráfico 2D ou 3D, precisamos diminuir as dimensões. O PCA reduz as 4 dimensões para 2 dimensões preservando ao máximo a variação, permitindo plotar os dados em um gráfico 2D, mantendo a distinção entre as espécies.

1.4 Activity PCA Example with the Iris data set

Vamos reduzir a dimensionalidade do dataset 'iris' lembrando que temos dados de quatro dimensões, o comprimento e a largura de sépalas e pétalas para quatro tipo de Iris (setosa, versicolor e virginica), com 150 observações.

FIGURA 8 – DATASET IRIS



The image shows a Jupyter Notebook interface with a file explorer at the top displaying 'Card_10_Aulaipy' and 'Aula_Using_PCA.py'. The notebook cell contains the following Python code:

```
1 print('hello world')
2
3 # importando bibliotecas
4 from sklearn.datasets import load_iris
5 from sklearn.decomposition import PCA
6 import matplotlib.pyplot as plt
7 from itertools import cycle
8
9 # importando dataset
10 iris = load_iris()
11
12 numSamples, numFeatures = iris.data.shape
13
14 print(f'{numSamples}\n{numFeatures}\n {list(iris.target_names)}')
15
16 # Chamando as funções
17
```

To the right, a terminal window titled 'C:\Windows\SYSTEM32\cmd.exe' shows the output of the code:

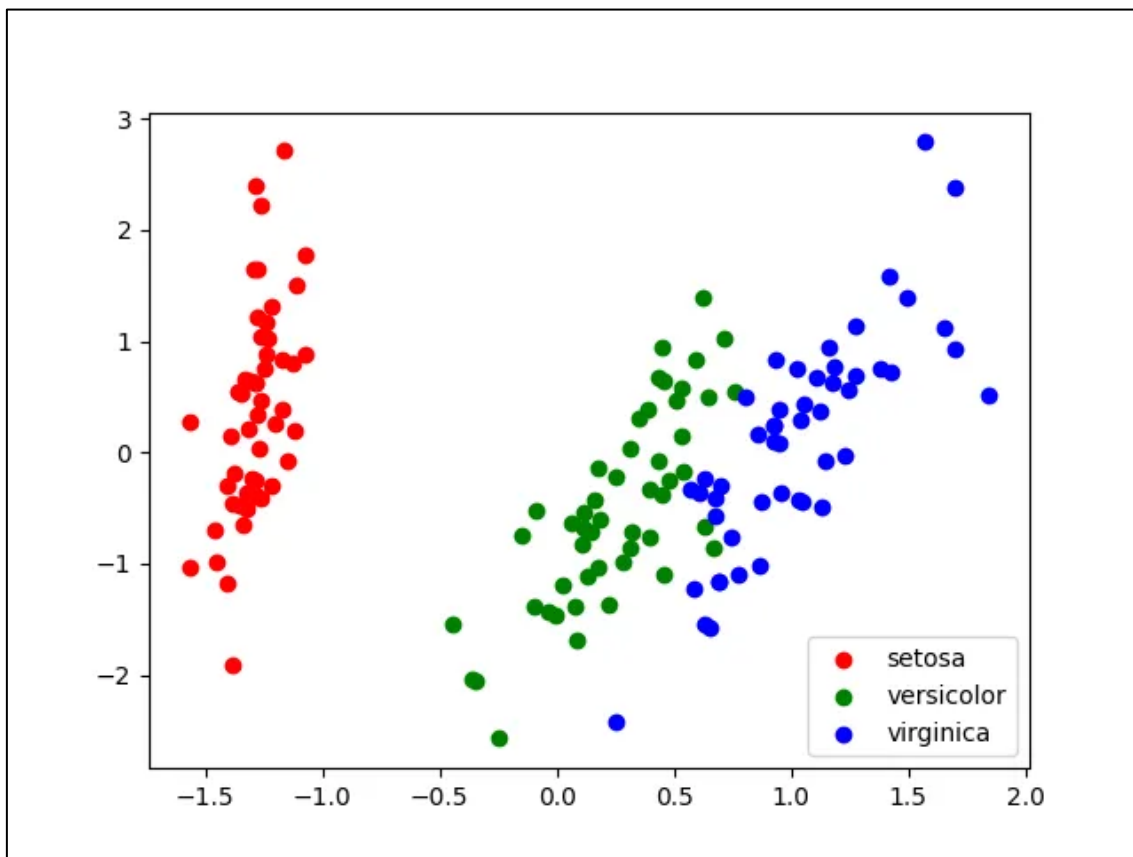
```
hello world
150
4
[np.str_('setosa'), np.str_('versicolor'), np.str_('virginica')]

-----
(program exited with code: 0)
Pressione qualquer tecla para continuar. . .
```

Elaborado pelo próprio autor, 2025.

Reduziremos a dimensionalidade dos dados para duas dimensões. Após isso precisamos ver o quanto de variância foi preservada para PC1 e PC2, nesse caso, 0.92 e 0.05, resultando numa variância total preservada de 0.97, isso significa que 97% da informação de quatro dimensões foi preservada em duas dimensões. Se a variância for muito baixa os dados vão apresentar muitos ruídos. Depois plotamos tudo em um gráfico.

FIGURA 9 – GRÁFICO COM DIMENSÕES REDUZIDAS



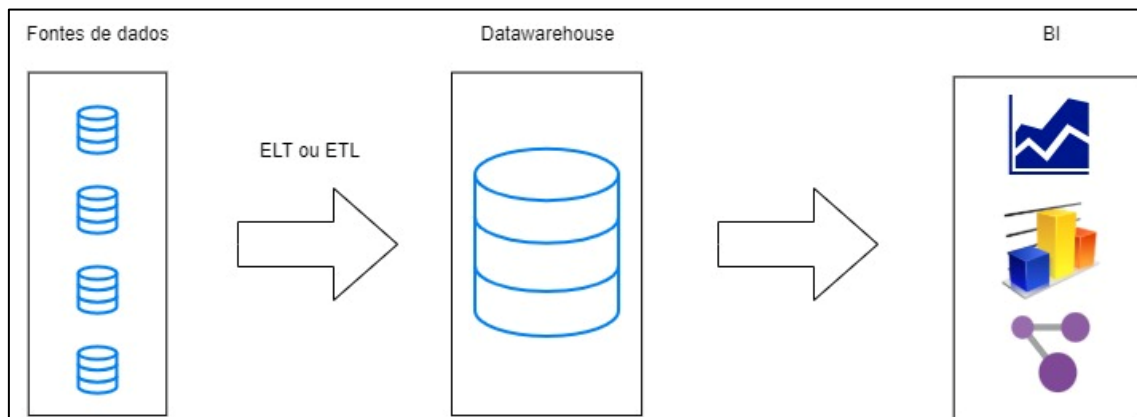
Elaborado pelo próprio autor, 2025.

1.5 Data Warehousing Overview ETL and ELT

Primeiro precisamos abordar o que é o DW (Data Warehouse), se trata de uma base de dados que centraliza dados de muitas fontes diferentes. Ele não centraliza todos os dados, mas apenas dados que são importantes para uma empresa ou organização. É muito utilizado pelo analista de BI (Business Intelligence) para criar relatórios. É geralmente construído através da linguagem SQL.

Departamentos inteiros são dedicados para manter o DW, primeiro porque é muito difícil manter os dados normalizados, por exemplo, como um dado se relaciona com o outro? Quais conjuntos de dados as pessoas precisam e por que? Segundo manter o fluxo de dados é um desafio contínuo, pois as fontes de dados podem mudar, ou a conexão com a internet pode cair, ou o site ou link pode mudar, etc. Terceiro, é muito difícil e custoso aumentar a escala.

FIGURA 10 – DATA WAREHOUSE



Elaborado pelo próprio autor, 2025.

Antes de chegarmos no ‘Data Warehouse, precisamos explorar a forma como é feita a importação dos dados das diferentes fontes de dados. Existem duas formas, ETL (Extract, Transform and Load) ou ELT (Extract, Load and Transform). A primeira forma é a mais tradicional, consiste em extrair os dados de forma periódica, depois transformar os dados no esquema necessário para o DW e finalmente carregar esses dados já estruturados. Em grandes conjuntos de dados, a etapa de transformar pode ser um grande problema, com isso surge uma pergunta, como lidar com o ‘big data’?

Para isso usamos a segunda forma, o ELT consiste em primeiro extrair, depois carregar os dados para o DW e aí sim realizarmos a etapa de transformação. Existem ferramentas que ajudam no processamento desses dados contidos no DW, o ‘Hadoop’ é uma delas, ele permite o processamento paralelo para massivos volumes de dados diretamente no DW.

1.6 Reinforcement Learning

Nós temos um agente que explora um espaço, esse agente vai aprender com os valores dos diferentes estágios e condições. Esses valores guiam o comportamento futuro do agente. A isso denominamos aprendizado por reforço. Uma implementação específica desse modo de aprendizado é o algoritmo ‘Q-learning’.

De forma simples, ele funciona da seguinte maneira, temos um conjunto de estados do ambiente (s), um conjunto de ações possíveis nesses estados (a) e um valor associado a cada par estado/ação (Q). Iniciamos com valores Q zerados, ao explorar o espaço, quando algo ruim acontece o valor de Q reduz, se algo bom acontecer o valor de Q aumenta.

Para o exemplo do jogo Pac-man, ele tem as seguintes opções dentro do jogo, ele pode bater na parede, pode morrer se encostar nos inimigos, ou caso nenhuma dessas opções aconteça, ele continua 'vivo'. Uma vez que o ambiente do jogo é dinâmico, não calculamos um passo de cada vez, temo que calcular todas as possibilidades antes, para isso usamos a seguinte fórmula $Q(s,a) += \text{discount} * (\text{reward}(s,a) + \max(Q(s')) - Q(s,a))$, onde s é o estado anterior e s' é o estado atual.

Com isso atingimos o problema da exploração, ele consiste em descobrir qual é a forma mais eficiente de explorar os possíveis estados. Uma abordagem simples é sempre escolher a ação com a maior recompensa, porém isso pode ser muito ineficiente, pois pode ignorar atalhos ou possibilidades de caminhos por não estarem com os valores mais altos na hora da decisão. Uma outra abordagem é inserir um termo ϵ , se um número aleatório for menor, não seguir o maior Q , mas sim uma ação aleatória, dessa forma a exploração nunca acontece, porém encontrar o valor de ϵ pode ser difícil.

Essa exploração tem um termo técnico, Processo de Decisão de Markov (Markov Decision Process - MDP), elas fornecem uma estrutura matemática para modelar a tomada de decisão em situações onde os resultados são parcialmente aleatórios e parcialmente controláveis pelo agente. Elas são um processo estocástico de controle em tempo discreto. Elas são uma forma matemática de descrever o que o algoritmo de 'Q-learning' faz.

Programação dinâmica, é o método de resolver problemas complexos quebrando em uma coleção de subproblemas simples, resolvendo cada um desses subproblemas e armazenando seus resultados. Da próxima vez que um subproblema acontece, ao invés de recalcular tudo novamente, apenas olhamos a solução computada, salvando tempo de computação em troca de um pouco de gasto com armazenamento.

No caso do pac-man, para criar um pac-man inteligente, primeiro precisamos explorar diferentes opções de forma semi-aleatória (ações), com diferentes condições (estados). Rastrear as recompensas e aprendizados associados a cada uma das opções (Q). Usar as informações armazenadas em futuras decisões.

1.7 Activity Reinforcement Learning & Q-Learning with Gym

Para praticar o aprendizado do algoritmos Q-learning, vamos explorar a biblioteca 'gym' nela conseguimos simula a seguinte situação, um táxi autônomo que precisa buscar passageiros em um local e entregar em outro local.

FIGURA 11 – TÁXI AUTÔNOMO



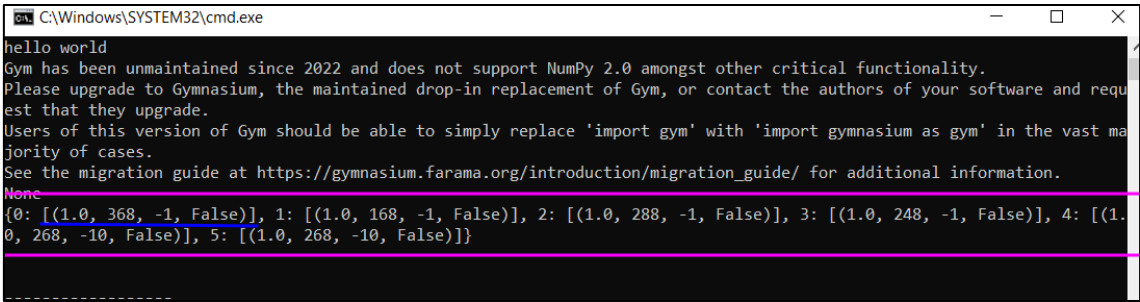
Elaborado pelo próprio autor, 2025.

Nesse caso, os pontos, amarelo, vermelho, verde e azul são onde pode haver 'pessoas' que o táxi precisa buscar ou locais que ele precisa deixar uma pessoa. A quantidade de possibilidade é calculada através do grid de 5x5, 4 possibilidades de destino e 5 de passageiros (4 nos pontos e 1 dentro do táxi), calculando, $5 \times 5 \times 4 \times 5 = 500$ possibilidades.

Para cada estado temos 6 possíveis ações, mover entre 4 direções, pegar um passageiro ou deixar um passageiro. O Q-learning, vai ter as seguintes opções deixar o passageiro no local correto (20 pontos), cada movimento com um passageiro dentro (-1 pontos) e buscar ou entregar em local incorreto (-10 pontos). Além disso, movimentos através da parede são ilegais.

Primeiro vamos definir a localização do táxi (2,3). O passageiro estará no ponto 2 e o destino no ponto 0. Após isso, imprimimos a lista de possíveis ações (5), ela é composta de 4 valores, a probabilidade de escolher uma opção, o próximo estado resultado daquela ação, a recompensa dessa ação e o indicativo de sucesso para deixar o passageiro.

FIGURA 12 – TÁXI AUTÔNOMO



Elaborado pelo próprio autor, 2025.

Precisamos agora treinar o modelo, para isso definimos uma taxa de aprendizado (0,1), um fator de desconto (0,6), fator de exploração (0,1) e a quantidade de épocas de treinamento (10000). Com isso podemos rodar nossa simulação e ver nosso modelo autônomo de taxi funcionar.

1.8 Understanding a Confusion Matrix

Uma matriz de confusão surge pois nem sempre a acurácia conta toda a ‘história’. Em um teste para doenças raras que possua acurácia de 99,9%, conseguiu isso ao ‘adivinhar’ ‘não’ a maior parte das vezes em que foi testado. Isso implica que precisamos entender verdadeiros/falsos positivos e negativos. Uma matriz de confusão consegue nos mostrar isso.

FIGURA 13 - MATRIZ DE CONFUSÃO

	Resultado Sim	Resultado Não
Predito Sim	Verdadeiro Positivo	Falso Positivo
Predito Não	Falso Negativo	Verdadeiro Negativo

Elaborado pelo próprio autor, 2025.

Se o resultado de um teste for ‘sim’ e o valor predito for ‘sim’ então é um ‘verdadeiro positivo’, se o valor predito for ‘não’ então será um ‘falso negativo’. O mesmo

vale para o contrário, se o resultado de um teste for ‘não’ e o valor predito for ‘sim’ então será um ‘falso positivo’, se o valor predito for ‘não’ então será um ‘verdadeiro negativo’.

1.9 Measuring Classifiers (Precision, Recall, F1, ROC, AUC)

Baseado em nossa matriz de confusão, nós temos alguns indicadores que conseguimos medir, o primeiro que vamos abordar é o ‘Recall’, é composto da seguinte equação:

$$\frac{\text{Verdadeiros positivos}}{\text{Verdadeiros positivos} + \text{falsos negativos}}$$

Ele possui outros nomes como ‘aka sensitive’, ‘true positive rate’ e ‘completeness’. Esse indicador é muito bom para quando se importa muito com falsos negativos (exemplo, detecção de fraude).

FIGURA 14 – DETECÇÃO DE FRAUDE

	Fraude	Não é fraude
Predito ‘É fraude’	5	20
Predito ‘Não é fraude’	10	100

Elaborado pelo próprio autor, 2025.

$$\frac{5}{(5 + 10)} = 33\% \text{ de 'recall'}$$

A segunda equação é a ‘precision’, muito parecida com a anterior, também é chamada de ‘aka correct positives’, representa a porcentagem de resultados relevantes. É uma boa escolha quando a métrica mais importante são os falsos positivos (exemplo, teste de drogas, detecção de doenças, etc).

$$\frac{\text{Verdadeiros positivos}}{\text{Verdadeiros positivos} + \text{falsos positivos}}$$

Usando o mesmo exemplo anterior:

$$\frac{5}{(5 + 20)} = 20\% \text{ de 'precision'}$$

Outras métricas, temos:

$$\text{Specificity} = \frac{\text{Verdadeiros negativos}}{\text{Verdadeiros negativos} + \text{falsos positivos}} = \text{'true negative rate'}$$

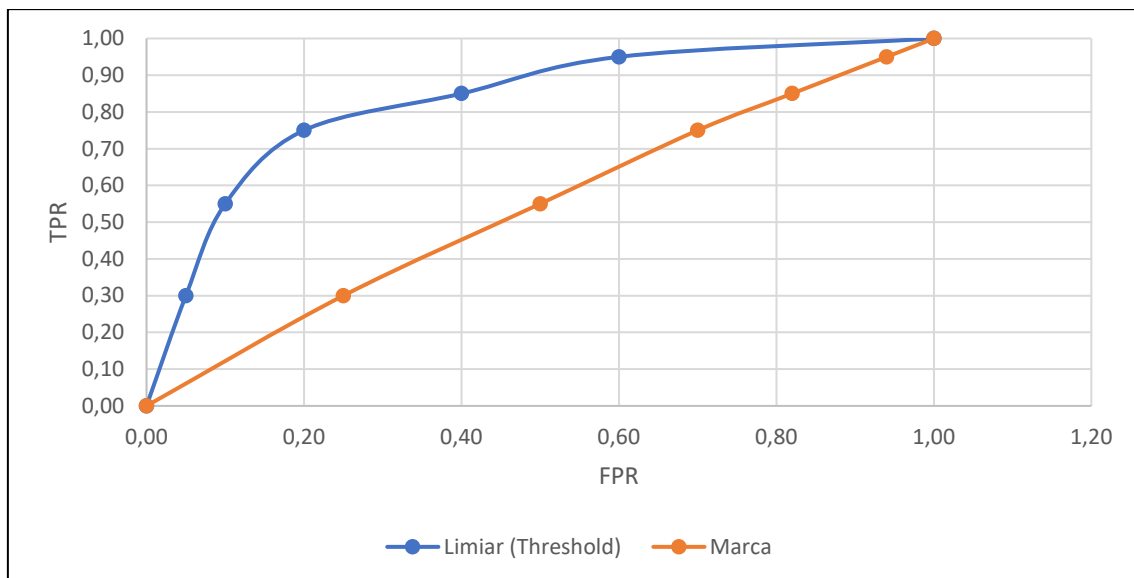
$$F1\ Score = \frac{2VP}{2VP + FP + FN}$$

OU

$$F1\ Score = 2 * \frac{Precision * recall}{Preciaion + recall}$$

Outro indicador é a ‘ROC Curve’, Curva Característica de Operação do Receptor, Gráfico da taxa de verdadeiros positivos (recall) vs. taxa de falsos positivos em várias configurações de limiar. Pontos acima da diagonal representam boa classificação (melhor que aleatória). A curva ideal seria apenas um ponto no canto superior esquerdo. Quanto mais "dobrada" em direção ao canto superior esquerdo, melhor.

FIGURA 15 – ROC CURVE



Elaborado pelo próprio autor, 2025.

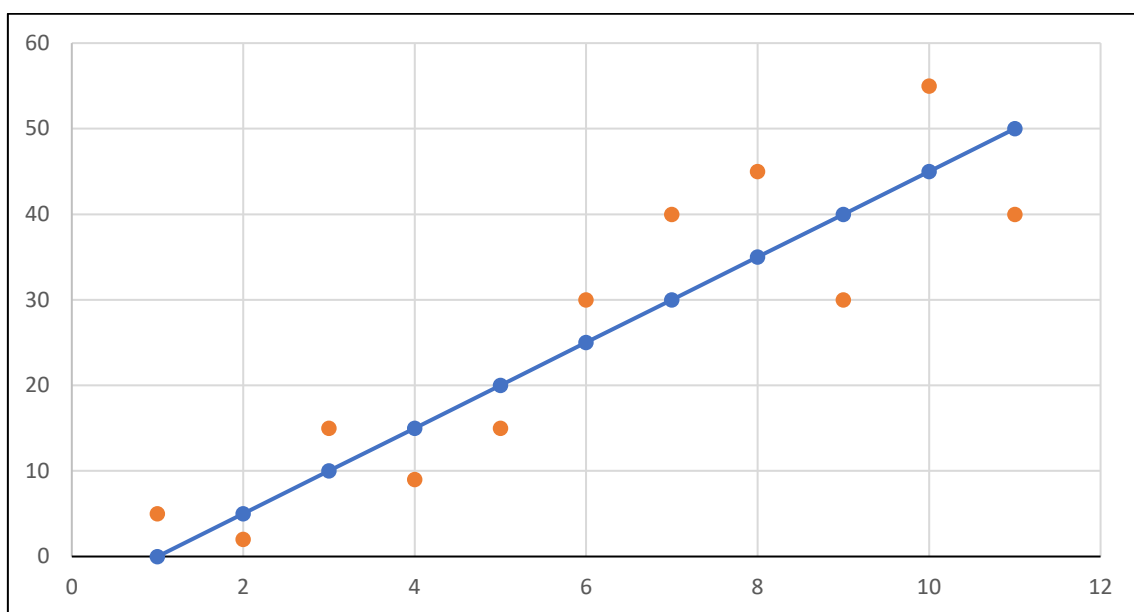
Também temos o indicador AUC, que é a área sob a curva Equivalente à probabilidade de que o classificador ranqueeie uma instância positiva aleatória acima de uma negativa aleatória. AUC-ROC de 0,5 indica um classificador inútil (equivalente a aleatoriedade) e 1,0 representa perfeição. É uma Métrica comum para comparação de classificadores.

2. Dealing with Real-World Data

2.1 Bias Variance Tradeoff

Quando tratamos de treinamento de máquina, dois conceitos são muito importantes, o viés e variância. O primeiro refere-se a distância da média dos seus valores preditos comparado com a resposta "real". O segundo, é o quão disperso os valores preditos estão da resposta "real".

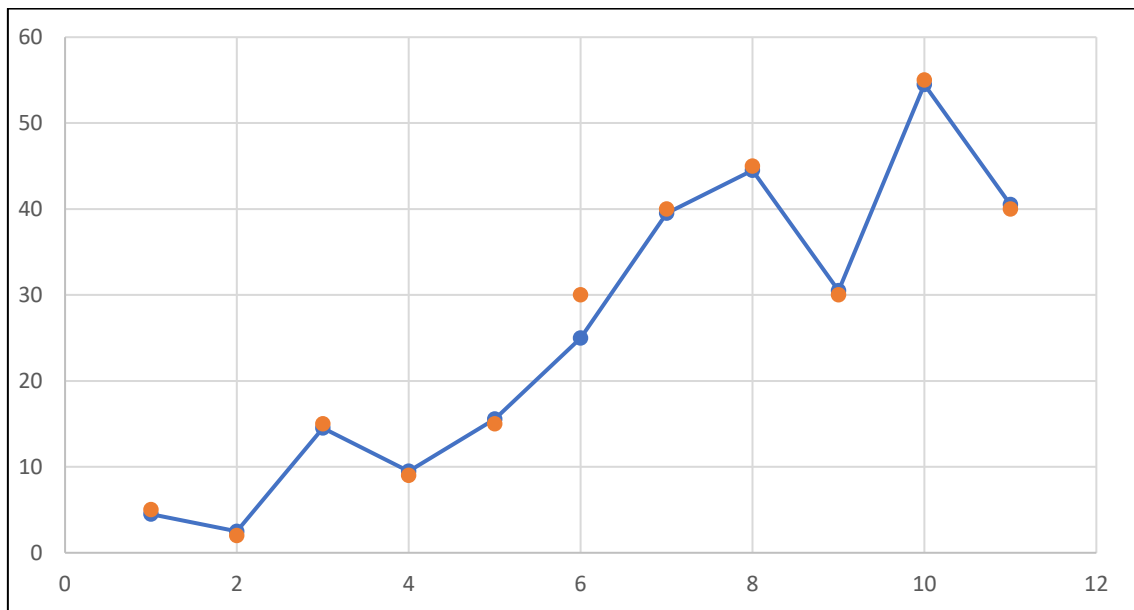
FIGURA 16 – EXEMPLO VIÉS, VARIÂNCIA



Elaborado pelo próprio autor, 2025.

Nesse exemplo, temos baixa variância e alto viés nesse caso podemos dizer que os dados são underfitting, ou seja, o modelo é muito simples não conseguindo capturar a relação entre os dados de entrada e saída. O resultado disso, é um desempenho ruim do modelo.

FIGURA 17 – EXEMPLO VIÉS, VARIÂNCIA



Elaborado pelo próprio autor, 2025.

Nesse exemplo, temos alta variância e baixo viés nesse caso podemos dizer que o modelo está overfitting, ou seja, o modelo é muito complexo se ajustando demais aos dados de treinamento, incluindo o ‘erro’. O resultado disso, é um desempenho ruim do modelo com os dados dos testes.

O que realmente importa é que tanto o viés como a variância contribuem para o erro do modelo, e o independente da complexidade do modelo, estamos sempre tentando diminuir o erro. A fórmula matemática para isso é: $Erro = Viés^2 + Variância$

2.2 Activity K-Fold Cross-Validation to avoid overfitting

Uma forma de evitar o overfitting é usar o método *k-fold cross validation* (Validação Cruzada K-fold). Ele consiste nos seguintes passos:

- Dividir seus dados em K segmentos randomicamente selecionados;
- Reservar um segmento como seus dados de teste;
- Realizar o treinamento ou a combinação do restante dos segmentos e mensurar sua performance contra os dados de teste;
- Repetir esse processo para cada segmento separando um segmento para teste;
- Calcular a média dos K valores de R quadrado.

A ideia é não usar todos os dados, dessa forma se garante que o modelo não irá se ajustar aos dados. Na prática, você precisa testar diferentes variações do seu modelo e medir a acurácia média usando Validação Cruzada K-Fold até encontrar o ponto ideal.

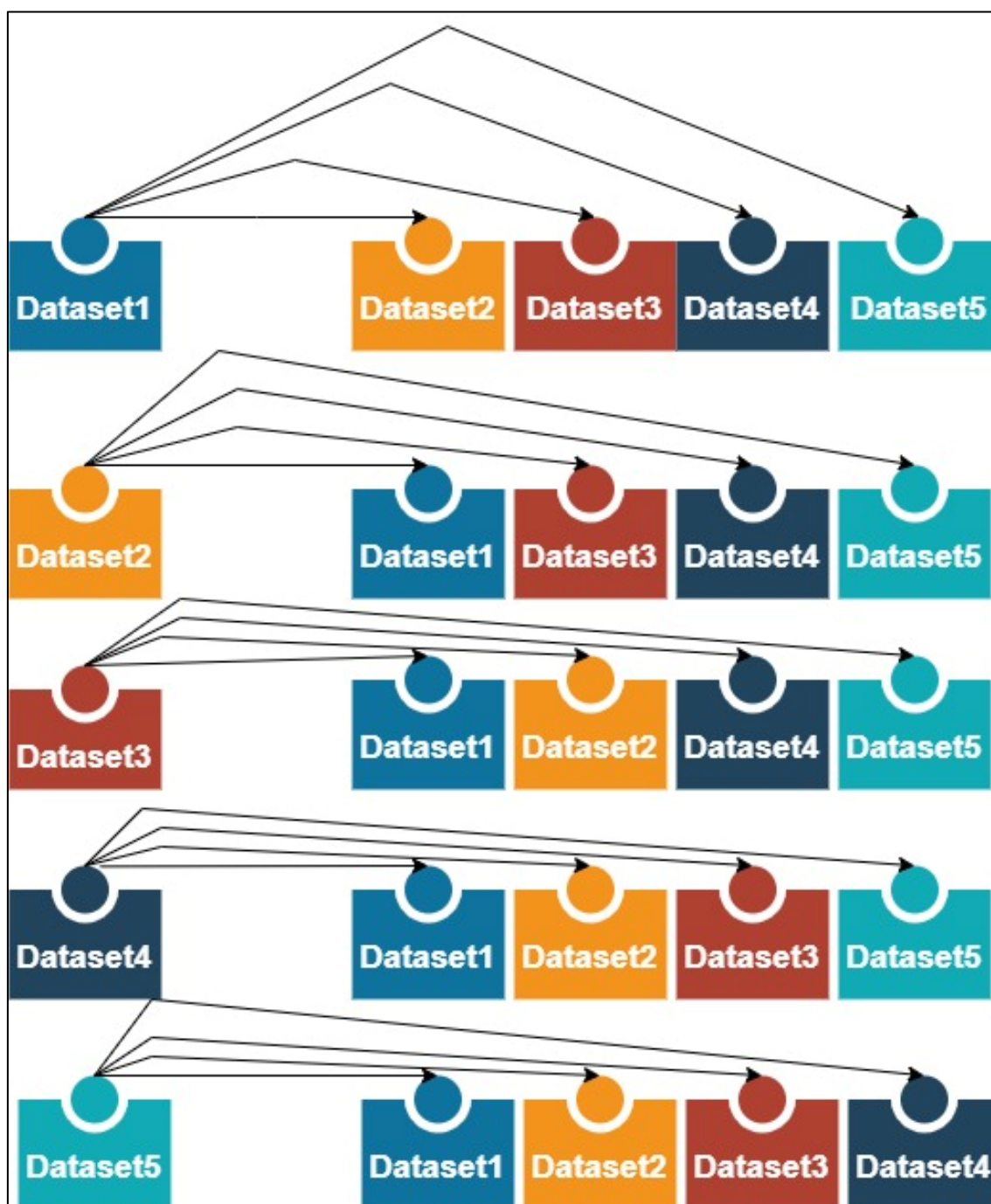
Como prática vamos usar os dados do dataset '*iris*'. Como já visto anteriormente, cada tipo de Iris tem uma pétala e sépala, o desafio é montar um modelo que consiga dizer a qual espécie de iris é a a pétala e sépala. O primeiro passo a seguir é dividir tantos os dados da sépala e da pétala em dados de treino e de teste. Depois escolhemos um modelo para treinar, neste caso foi usado o modelo linear. Após isso medimos a sua performance com os dados de teste, com uma boa performance seguimos para a validação k-fold.

Para essa validação, foi usado $k = 5$, ou seja, o dataset é dividido em 5 partes iguais, e o processo ocorre da seguinte maneira: treina-se o modelo com 4 partes e testa a performance dele com 1 parte, faz esse processo de novo, mas agora usando aquela parte de validação para o treino e separando uma outra parte para validar, repete-se o processo até validar com todas as partes do dataset, ao fim mede-se a performance do treino para cada dataset e depois a performance média.

FIGURA 18 – DATASET E K-FOLD



FIGURA 19 – FUNCIONAMENTO DO K-FOLD(5)



Elaborado pelo próprio autor, 2025.

FIGURA 20 – INTERPRETANDO RESULTADO

```

1 import numpy as np
2 from sklearn.model_selection import train_test_split
3 from sklearn.model_selection import cross_val_score
4 from sklearn import datasets
5 from sklearn import svm
6
7 iris = datasets.load_iris()
8
9 # A single train/test split is made easy with the train_test_split function in sklearn
10
11 # Split the Iris data into train/test data sets with 40% reserved for testing
12 X_train, X_test, Y_train, Y_test = train_test_split(iris.data, iris.target, test_size=0.4, random_state=0)
13
14 print(X_train, '\n')
15 print(X_test, '\n')
16 print(Y_train, '\n')
17 print(Y_test, '\n')
18
19 # Build an SVC model for predicting iris classifications using training data
20 clf = svm.SVC(kernel='linear', C=1).fit(X_train, Y_train)
21
22 print(clf, '\n')
23
24 # Now measure its performance with the test data
25 clf.score(X_test, Y_test)
26 print(clf.score(X_test, Y_test), '\n')
27
28 # K-Fold cross validation is just as easy: let's use a K of 5:
29
30 # We give cross_val_score a model, the entire data set and its "real" values,
31 scores = cross_val_score(clf, iris.data, iris.target, cv=5)
32
33 # Print the accuracy for each fold:
34 print(scores, '\n')
35
36 # And the mean accuracy of all 5 folds:
37 print(scores.mean(), '\n')

```

```

C:\Windows\SYSTEM32\cmd.exe
[5.5 2.4 3.8 1.1]
[6.3 2.7 4.9 1.8]
[6.3 2.8 5.1 1.5]
[4.9 2.5 4.5 1.7]
[6.3 2.5 5. 1.9]
[7. 3.2 4.7 1.4]
[6.5 3. 5.2 2. ]]

[1 0 2 1 1 1 1 2 0 0 2 1 0 0 1 0 2 1 0 1 2 1 0 2 2 2 2 0 0 2 2 0 2 0 2 2 0
0 2 0 0 0 1 2 2 0 0 0 1 1 0 0 1 0 2 1 2 1 0 2 0 2 0 0 2 0 2 1 1 1 2 2 1 1
0 1 2 2 0 1 1 1 1 0 0 0 2 1 2 0]

[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
1 1 1 2 0 2 0 0 1 2 2 2 2 1 2 1 1 2 2 2 1 2]

SVC(C=1, kernel='linear')
0.9666666666666667

[0.96666667 1. 0.96666667 0.96666667 1. ]

```

Elaborado pelo próprio autor, 2025.

O resultado do k-fold para 5 datasets, foi 96% de performance para o primeiro dataset, 100% de performance para o segundo, 96% para o terceiro, 96% para o quarto e 100% para o quinto, resultando em uma média de 98% de performance. Como comparação, se tentarmos medir a performance de um modelo polinomial, mesmo usando k-fold ele entregue menos performance que o modelo linear, ou seja, mais complexidade não significa mais performance.

FIGURA 21 – LINEAR X POLINOMIAL

```

1 import numpy as np
2 from sklearn.model_selection import train_test_split
3 from sklearn.model_selection import cross_val_score
4 from sklearn import datasets
5 from sklearn import svm
6
7 iris = datasets.load_iris()
8
9 # A single train/test split is made easy with the train_test_split function in sklearn
10
11 # Split the Iris data into train/test data sets with 40% reserved for testing
12 X_train, X_test, Y_train, Y_test = train_test_split(iris.data, iris.target, test_size=0.4, random_state=0)
13
14 print(X_train, '\n')
15 print(X_test, '\n')
16 print(Y_train, '\n')
17 print(Y_test, '\n')
18
19 # Build an SVC model for predicting iris classifications using training data
20 clf = svm.SVC(kernel='linear', C=1).fit(X_train, Y_train)
21
22 print(clf, '\n')
23
24 # Now measure its performance with the test data
25 clf.score(X_test, Y_test)
26 print(clf.score(X_test, Y_test), '\n')
27
28 # K-Fold cross validation is just as easy: let's use a K of 5:
29
30 # We give cross_val_score a model, the entire data set and its "real" values,
31 scores = cross_val_score(clf, iris.data, iris.target, cv=5)
32
33 # Print the accuracy for each fold:
34 print('Usando modelo linear e k(5):', scores, '\n')
35
36 # And the mean accuracy of all 5 folds:
37 print('Média de modelo linear e k(5):', scores.mean(), '\n')
38
39 # Com o resultado bom vamos reconstruir usando uma função polinomial
40
41 # Build an SVC model for predicting iris classifications using training data
42 clf = svm.SVC(kernel='poly', C=1).fit(X_train, Y_train)
43
44 scores = cross_val_score(clf, iris.data, iris.target, cv=5)
45 print('Usando modelo polinomial e k(5):', scores, '\n')

```

```

C:\Windows\SYSTEM32\cmd.exe
[7. 3.2 4.7 1.4]
[6.5 3. 5.2 2. ]]

[1 0 2 1 1 1 1 2 0 0 2 1 0 0 1 0 2 1 0 1 2 1 0 2 2 2 2 0 0 2 2 0 2 0 2 2 0
0 2 0 0 0 1 2 2 0 0 0 1 1 0 0 1 0 2 1 2 1 0 2 0 2 0 0 2 0 2 1 1 1 2 2 1 1
0 1 2 2 0 1 1 1 1 0 0 0 2 1 2 0]

[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
1 1 1 2 0 2 0 0 1 2 2 2 2 1 2 1 1 2 2 2 1 2]

SVC(C=1, kernel='linear')
Usando modelo linear sem k-fold: 0.9666666666666667
Usando modelo linear e k(5): [0.96666667 1. 0.96666667 0.96666667 1. ]
Média de modelo linear e k(5): 0.9800000000000001

Usando modelo polinomial e k(5): [0.96666667 1. 0.96666667 0.96666667 1. ]
Média de modelo polinomial e k(5): 0.9800000000000001
Usando modelo polinomial sem k-fold: 0.9
(program exited with code: 0)

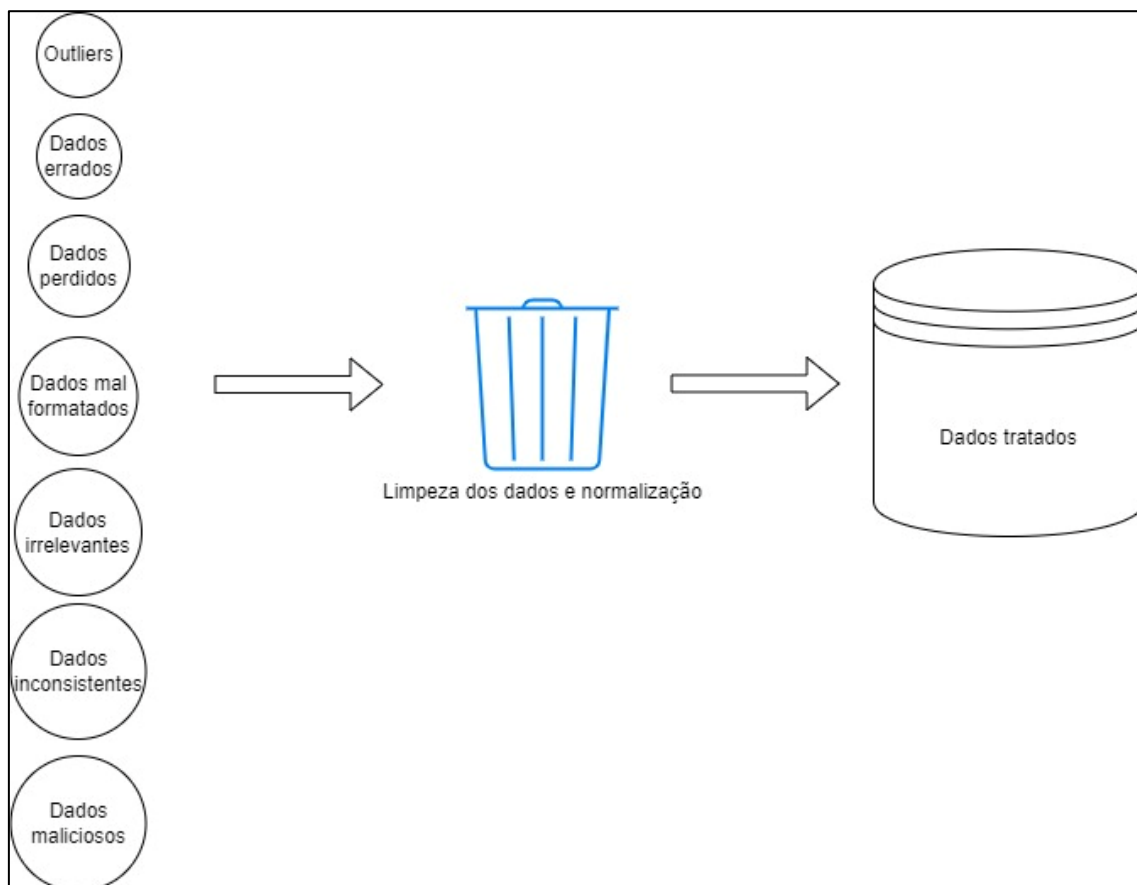
```

Elaborado pelo próprio autor, 2025.

2.3 Data Cleaning and Normalization

Neste tópico abordaremos uma verdade inconveniente sobre trabalhar como cientista de dados, que é limpar os dados. Se gasta muito tempo com esse tipo de trabalho, ajustando outliers, dados perdidos, maliciosos, errados, irrelevantes, inconsistentes, mal formatados e etc. E isso é uma parte muito importante do trabalho, pois, dados bons entregam bons resultados, dados ruins entregam maus resultados.

FIGURA 22 – DATA CLEANING E NORMALIZATION



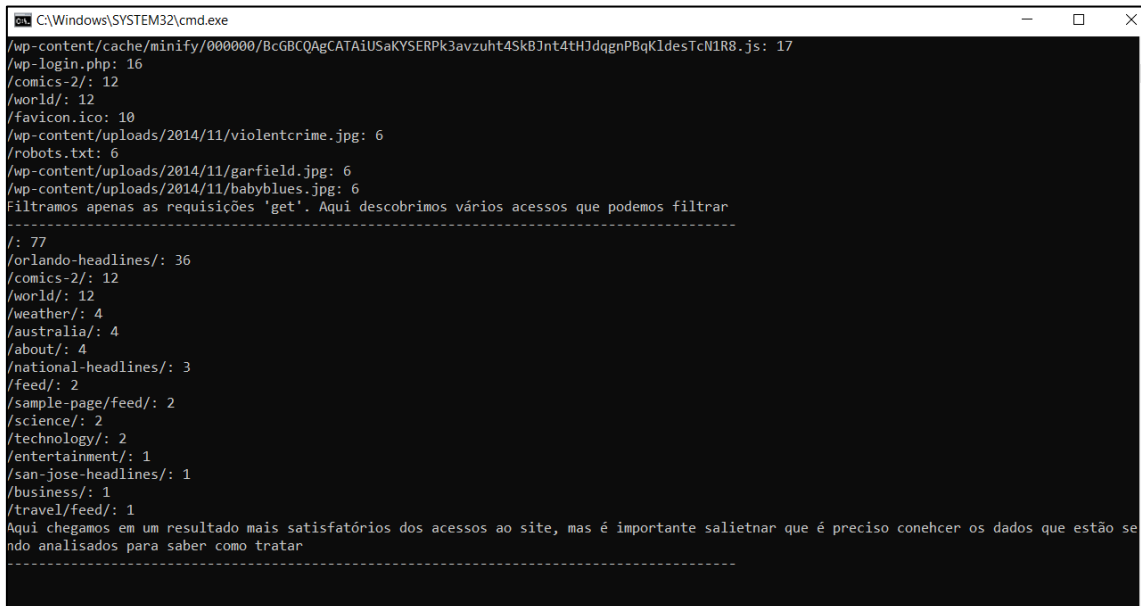
Elaborado pelo próprio autor, 2025.

2.4 Activity Cleaning web log data

Aqui o objetivo é analisar os logs de acessos a um site. Para isso é importante entender muito bem o dataset que estamos analisando, nesse caso, sabemos que uma requisição comum tem tamanho de '3 itens'. Então para tratarmos, primeiro ignoramos os dados com mais de 3 itens, em seguida fazemos a análise dos resultados e vamos refinando passo-a-passo.

Ao fim, como estamos tratando de requisições HTTP, selecionamos apenas os itens que fazem a requisição 'GET' conseguimos refinar nosso resultado mais próximo daquilo que queremos analisar, a quantidade de acessos ao site.

FIGURA 23 – ANÁLISE DE LOG



```
C:\Windows\SYSTEM32\cmd.exe
/wp-content/cache/minify/000000/BcGBCQAgCATAiUSaKYSERPk3avzuht4SkBJnt4tHJdqgnPBqKlidesTcN1R8.js: 17
/wp-login.php: 16
/comics-2/: 12
/world/: 12
/favicon.ico: 10
/wp-content/uploads/2014/11/violentcrime.jpg: 6
/robots.txt: 6
/wp-content/uploads/2014/11/garfield.jpg: 6
/wp-content/uploads/2014/11/babyblues.jpg: 6
Filtramos apenas as requisições 'get'. Aqui descobrimos vários acessos que podemos filtrar
-----
/: 77
/orlando-headlines/: 36
/comics-2/: 12
/world/: 12
/weather/: 4
/australia/: 4
/about/: 4
/national-headlines/: 3
/feed/: 2
/sample-page/feed/: 2
/science/: 2
/technology/: 2
/entertainment/: 1
/san-jose-headlines/: 1
/business/: 1
/travel/feed/: 1
Aqui chegamos em um resultado mais satisfatório dos acessos ao site, mas é importante salientar que é preciso conhecer os dados que estão se
ndo analisados para saber como tratar
-----
```

Elaborado pelo próprio autor, 2025.

2.5 Normalizing numerical data

Aqui vamos abordar a normalização de dados numéricos, primeiro, pensemos, se o seu modelo é baseado em vários atributos numéricos, eles são comparáveis? Atributos do tipo 'idade' variam de 0 a 100, enquanto o atributo 'rendas' pode variar de 0 a bilhões.

Alguns modelos podem não funcionar de forma satisfatória quando diferentes atributos estão em escalas muito diferentes, isso faz com que alguns atributos tenham mais peso que os outros. Isso pode implicar em viés para o modelo, além disso, se houver viés nas variáveis isso impactará no modelo.

A implementação de PCA do Scikit-learn tem uma opção "whiten" (branquear) para normalizar os dados. O Scikit-learn possui um módulo de pré-processamento com funções úteis de normalização (normalize) e escala (scale). Importante, se seus dados são "sim" e "não", é preciso converter para "1" e "0".

É importante ler a documentação, a maioria das técnicas de mineração de dados e aprendizado de máquina funcionam bem com dados brutos e não normalizados, mas isso

precisa sempre ser verificado, e depois que encontrar o resultado é importante voltar para as escalas de valor original.

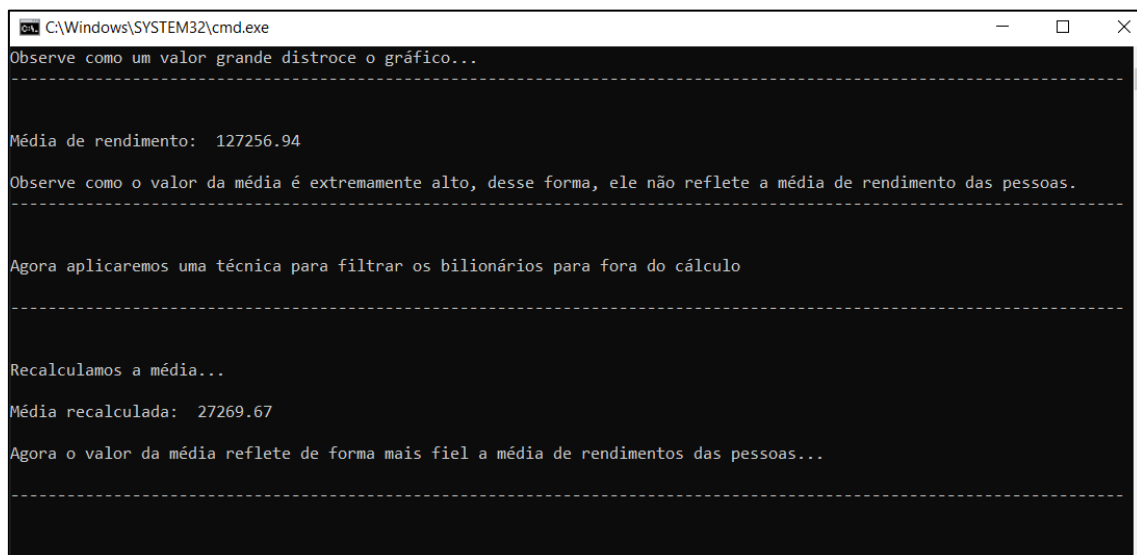
2.6 Activity Detecting outliers

Muitas vezes, ao analisar dados do mundo real, não vamos encontrar outliers apenas de ataques ou bots, vamos encontrar usuários que simplesmente tiveram aquele comportamento inesperado e a forma como você irá tratar isso é importante.

Às vezes é apropriado remover outliers dos seus dados de treinamento. Faça isso com responsabilidade! Entenda o porquê de estar fazendo isso. Por exemplo: na filtragem colaborativa, um único usuário que avalia milhares de filmes pode ter um grande efeito nas avaliações de todos os outros. Isso pode não ser desejável. Outro exemplo: em dados de web logs, outliers podem representar bots ou outros agentes que devem ser descartados. Mas se alguém realmente quer a renda média dos cidadãos dos EUA, por exemplo, não descarte os bilionários, eles são parte desse cálculo.

Para encontrar outliers podemos usar o desvio-padrão como uma forma de classificar-los. É importante definir um limiar onde a partir de um determinado ponto será considerado outlier. Qual ponto? Isso precisa ser testado porque varia para cada dataset e situação.

FIGURA 24 – MÉDIA COM E SEM OUTLIER



Elaborado pelo próprio autor, 2025.

2.7 Feature Engineering and the Curse of Dimensionality

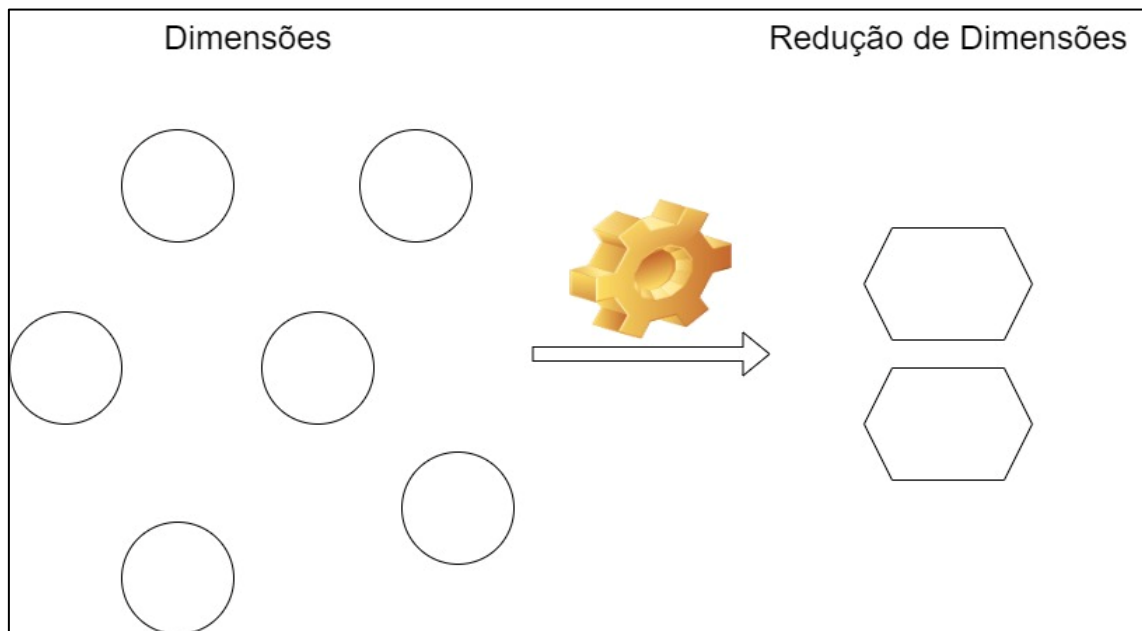
A feature engineering é a arte de aplicar seu conhecimento sobre os dados, e o modelo, para criar melhores features (características) para treinar seu modelo. É importante sempre definir quais features irá usar, essa escolha vai se dar baseado naquilo que estamos tentando prever.

Depois disso, é importante definir se é necessário transformar essas features, pois a escala dos dados selecionados pode impactar os resultados do modelo, logo, em algumas situações é preciso normalizar os dados primeiro. Anexo a isso, é importante definir como irá lidar com dados faltantes, se vai remove-lo ou substituir por algum valor. Não se pode simplesmente inserir dados brutos e esperar bons resultados, ter esse sentimento, é a arte do aprendizado de máquina é onde a expertise é aplicada

Concomitante a isso, temos a maldição da dimensionalidade, que consistem em ter muitas features, isso causa o problema de ter dados muito esparsos, porque cada features é uma nova dimensão e esses dados podem ficar muito distantes um dos outros. Grande parte da engenharia de features é selecionar apenas as características mais relevantes para resolver o problema em questão.

Conhecimento e domínio dos dados que estão sendo analisados é parte fundamental desse processo. Técnicas não supervisionadas de redução de dimensionalidade também podem ser usadas para reduzir as dimensões dos dados. Exemplo, PCA (Análise de Componentes Principais) e K-Means (Agrupamento K-Means).

FIGURA 25 – REDUÇÃO DE DIMENSIONALIDADE



Elaborado pelo próprio autor, 2025.

2.8 Imputation Techniques for Missing Data

Como dito no tópico anterior, é importante definir como tratar dados faltantes, para isso, é importante compreender algumas técnicas, a primeira e mais comum é a técnica da média. Ela consiste em substituir valores faltantes pela média dos valores restantes da coluna (só serve para coluna). Ela é rápida e fácil de ser feita e não afeta o tamanho do conjunto de dados. Quando há presença de outliers é recomendado usar a mediana no lugar da média.

A desvantagem dessa técnica é que ela pode ser pouco eficaz, pois funciona apenas a nível de coluna, ignorando correlação entre característica (exemplo, peso e altura, o peso pode ter correlação com a altura e preencher os dados faltantes do peso com a média pode ser um erro). Além disso, ela não pode ser usada em características categóricas e pode não ser muito precisa.

Outra técnica importante de usar, é a exclusão de dados faltantes, mas importante, apesar de simples, lembre-se, essa exclusão não pode enviesar seus dados e caso precise realizar uma análise rápida. No entanto, essa não é uma abordagem ideal, use-a apenas quando for necessário.

Pode ser utilizado o aprendizado de máquina para tratar esses valores vazios. Pode ser usado o KNN (K-Nearest Neighbors), para encontrar K linhas "mais próximas" (mais

similares) e calcule a média dos seus valores, mas só pode ser usado para dados numéricos. Também poderia usar o Deep Learning, para construir um modelo de aprendizado de máquina para colocar dados para o seu modelo de aprendizado de máquina, funciona bem para dados categóricos, a única desvantagem é o tempo de desenvolvimento. E por último a regressão, encontre relações lineares ou não lineares entre a característica faltante e outras características.

Apesar dessas técnicas é importante salientar que a busca por mais informações ou captar mais dados é imprescindível. As vezes um período de tempo, que está sendo analisado nos seus dados, tem muitos dados faltantes, mas logo em períodos seguintes eles podem estar completos.

2.9 Handling Unbalanced Data Oversampling, Undersampling, and SMOTE

Outro problema muito comum nos dados de vida real, são o desbalanceamento dos dados, ou seja, uma grande discrepância entre os casos ‘positivos’ e ‘negativos’, por exemplo, esse desbalanceamento é uma forma de detectar fraudes, uma vez que ao serem raras, quando acontecem são detectáveis. Lembre-se o termo ‘positivo’ não significa algo bom, apenas que o objeto de estudo testou positivo/negativo para sua hipótese.

O desbalanceamento é um problema para as redes neurais, com isso, surgiu uma técnica para lidar com isso, ‘oversampling’, ela consiste em aumentar de forma artificial o número das classes minoritárias, criando cópias de amostrar existentes ou gerando novas amostras fake.

O contrário também dá para ser feito, ao invés de criar mais amostras positivas, podemos remover as amostras negativas. A questão é que descartar dados nunca é uma boa solução, afinal é possível informação que estamos perdendo. Esse processo é bom quando esteja tentando evitar problema de escala com ‘big data’.

Outra técnica importante é o SMOTE (Técnica de Sombreamostragem Sintética de Minorias), que consiste em roda o algoritmo de k-vizinhos nos valores da classe minoritária e usar a média dos vizinhos para preencher os dados faltantes. Ela é melhor que apenas a sobreamostragem.

Em casos onde estamos fazendo previsão sobre uma classificação (fraude / não fraude), você tem um limiar de probabilidade a partir do qual algo é sinalizado como caso positivo, se seus dados possuem muitos falsos positivos, uma maneira de corrigir isso é

simplesmente aumentar esse limiar. O efeito disso é reduzir falsos positivos, mas pode resultar em mais falsos negativos.

2.10 Binning, Transforming, Encoding, Scaling, and Shuffling

Seguindo na exploração das técnicas aplicadas aos dados relacionados aos modelos de IA, temos a discretização em intervalos, ela consiste em agrupar observações em intervalos com base em faixas de valores. Por exemplo, se queremos estimar a idade das pessoas colocamos todas as pessoas de 20-29 anos em uma classificação, 30 – 39 anos em outra, etc. É possível fazer essa distribuição por quartis, isso garante que todos os intervalos terão tamanho igual.

Outro ponto importante é a transformação dos dados, é possível aplicar uma função a uma característica para torná-la mais adequada para o treinamento. Dados que possuem uma tendência exponencial podem se beneficiar de uma transformação logarítmica. Isso permite o aprendizado de funções superlineares e sublineares.

Codificação, consiste em transformar os dados em uma nova representação exigida pelo modelo. Temos a codificação one-hot (one-hot encoding), ela cria categorias binárias para cada categoria. Esse binário recebe o valor 1 e todos os outros recebem 0. É muito comum em aprendizado profundo (deep learning), onde as categorias são representadas por "neurônios" de saída individuais.

E por último, temos o escalonamento / normalização, ela consiste em colocar todos os atributos numéricos do seu modelo na mesma escala, alguns modelos preferem que os dados das características tenham distribuição normal. A maioria dos modelos exige que os dados das características sejam pelo menos escalonados para valores comparáveis. Caso contrário, características com magnitudes maiores terão mais peso do que deveriam.

Conclusões

A primeira etapa do conteúdo focou em explicar o algoritmo de k-means e como ele classifica os clusters através da distância entre eles. Depois focou na maldição da dimensionalidade e como conseguimos reduzir o grande numero de variáveis numéricas de uma análise, usando menos dados e entregando um resultado confiável.

Depois exploramos o treinamento por reforço e o uso do algoritmo q-learning, apesar de o código do exemplo ser antigo, foram necessárias algumas alterações no código para conseguir rodar de forma satisfatória. Depois exploramos o tópico da matriz

de confusão para escolhermos para qual lado nosso modelo preferencialmente irá ‘errar’ e como vai ‘errar’. Através de algumas métricas aprendemos como medir isso.

Na segunda etapa, exploramos como viés e variância podem afetar um modelo com problemas de underfitting e overfitting. Após isso foi apresentado como usar o k-fold com validação cruzada para evitar o problema de overfitting, quebrando seus dados de treino em cinco partes e calculando o modelo a partir de quatro partes e comparando com a parte separada, fazendo isso para cada um dos itens e no fim calculando a média da performance.

Depois foram abordados tópicos referentes a importância de tratar, normalizar os dados e detectar outliers antes de inserir no modelo para treino, pois isso pode criar um viés nos resultados. Aplicamos técnicas para tratar dados faltantes na base de dados e como cada uma delas impacta no modelo. E por últimos tópicos referentes a desbalanceamento de dados (alta amplitude), técnicas de discretização, codificação e escalonamento de dados.

Referencias

As referências foram apenas os vídeos da atividade.