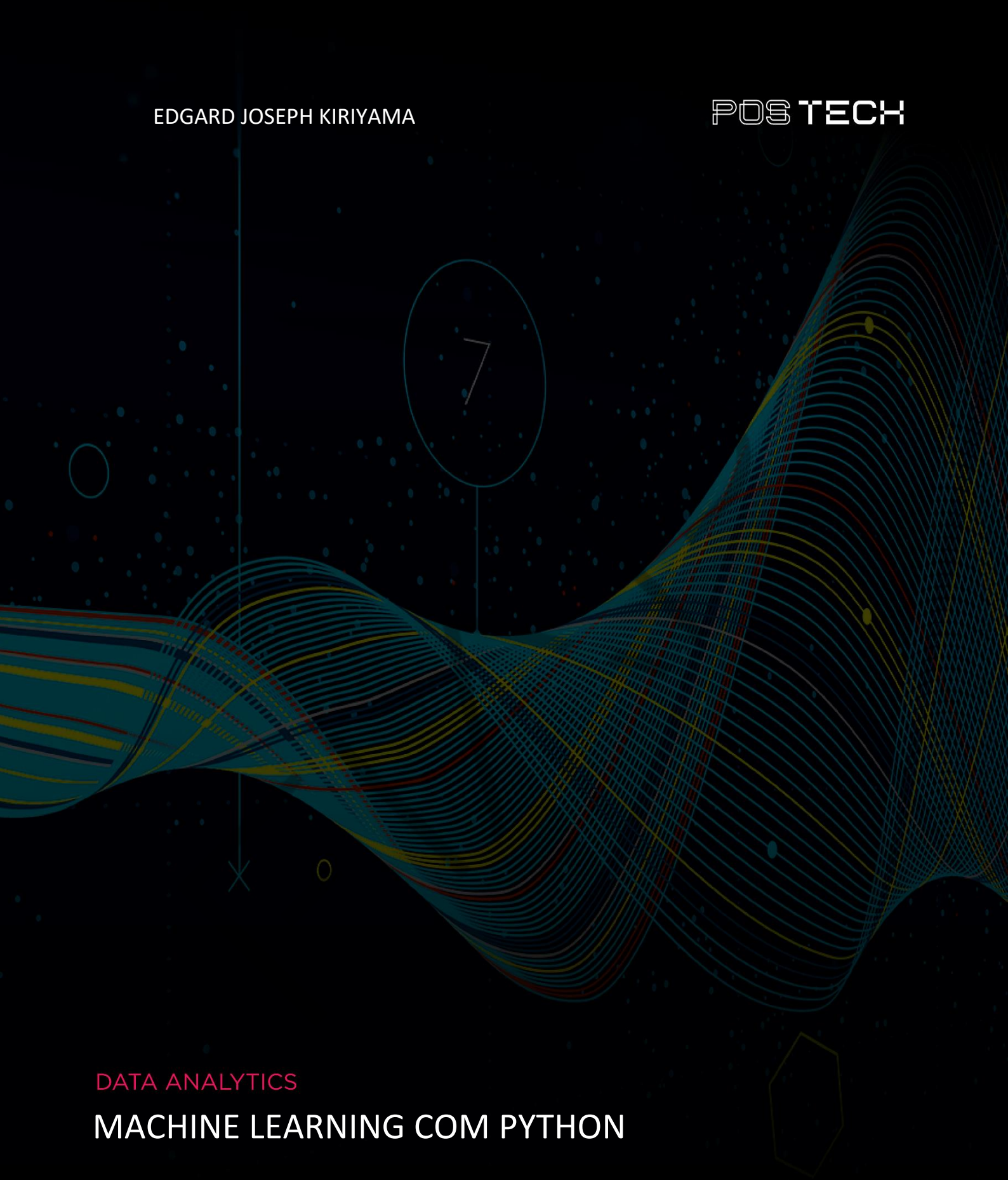
EDGARD JOSEPH KIRIYAMA

PÓS TECH



DATA ANALYTICS

MACHINE LEARNING COM PYTHON

AULA 03

SUMÁRIO

O QUE VEM POR AÍ? 3
HANDS ON 4
SAIBA MAIS 5
O QUE VOCÊ VIU NESTA AULA? 13
REFERÊNCIAS 14

O QUE VEM POR AÍ?

Olá, jovem analytic!

Você está na quarta disciplina do curso e aqui começa mais um ponto importantíssimo da jornada em que você está trilhando.

Na aula passada falamos sobre Análise Exploratória de Dados e como já podemos ter alguns insights quando visualizamos na nossa base de dados, tanto por dataframes quanto por gráficos.

Chegamos em outro momento importante de nossa jornada: Feature Engineering!

Este é o momento de aprimorarmos e elevarmos o nosso nível! Vamos te mostrar o quão importante é realizar a limpeza do dataset, renomear e classificar a nossa base de dados da melhor maneira para ser manipulável e preparar a base para que o nosso conjunto de dados seja bem estruturado para que os nossos modelos de Machine Learning consigam fazer bem a sua função.

Vamos começar a entender como esta ferramenta pode nos ajudar no nosso dia a dia! Bora lá?

HANDS ON

Agora, chegou o momento de ver, na prática, como começar a importar os nossos dados e trabalhar com eles via programação. A ideia é não se limitar apenas ao código explícito no hands on, então é sempre bom procurar a documentação das bibliotecas, explorar novas funcionalidades e muito mais!

SAIBA MAIS

A Feature Engineering é o processo de transformar os dados brutos em features, ou seja, em representações numéricas que melhor representem os dados e que serão aproveitados da melhor maneira possível nos modelos de Machine Learning (ML).

Claro que esta etapa só irá ocorrer depois que vocês executarem a etapa de Análise Exploratória dos Dados – EDA. Sem executar esta etapa, o que falaremos agora não fará sentido, pois vocês já identificaram que é necessário fazer a limpeza de dados nulos ou duplicados, transformar os tipos de dados, padronizar os dados para que os modelos possam executar seus algoritmos de forma correta, entre outros.

Quem já trabalhou com bases reais, percebeu que geralmente as bases não vêm da forma adequada. Sempre é necessário um mega trabalho para que a base, ao final, fique bem próxima para que os modelos de ML executem seus algoritmos de forma correta.

Vamos trabalhar com uma base real?

Então, iremos utilizar os dados de COVID de um estudo executado pelo IBGE – Instituo Brasileiro de Geografia e Estatística, o PNAD-COVID.

"O PNAD-COVID objetiva estimar o número de pessoas com sintomas referidos associados à síndrome gripal e monitorar os impactos da pandemia da COVID-19 no mercado de trabalho brasileiro" (IBGE, 2020).

O questionário se divide em duas partes, sendo uma direcionada a questões de saúde, especificamente sobre sintomas associados à síndrome gripal e outra a questões de trabalho. Nas questões de saúde, investiga-se a ocorrência de alguns dos principais sintomas da COVID19 no período de referência da pesquisa, considerando-se todos os moradores do domicílio.

Vamos visualizar os dados para entendermos um pouco a complexidade?

```
[1] import pandas as pd

[2] df = pd.read_csv('PNAD_COVID_112020.csv')

    df
```

	Ano	UF	CAPITAL	RM_RIDE	V1008	V1012	V1013	V1016	Estrato	UPA	...	F001	F0021	F0022
0	2020	11	11.0	NaN	1	4	11	7	1110011	110015970	...	1.0	NaN	NaN
1	2020	11	11.0	NaN	1	4	11	7	1110011	110015970	...	1.0	NaN	NaN
2	2020	11	11.0	NaN	1	4	11	7	1110011	110015970	...	1.0	NaN	NaN
3	2020	11	11.0	NaN	1	4	11	7	1110011	110015970	...	1.0	NaN	NaN
4	2020	11	11.0	NaN	2	1	11	7	1110011	110015970	...	1.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
157071	2020	29	NaN	NaN	2	3	11	7	2954010	290018812	...	5.0	NaN	NaN
157072	2020	29	NaN	NaN	5	4	11	7	2954010	290018812	...	1.0	NaN	NaN
157073	2020	29	NaN	NaN	5	4	11	7	2954010	290018812	...	1.0	NaN	NaN
157074	2020	29	NaN	NaN	5	4	11	7	2954010	290018812	...	1.0	NaN	NaN
157075	2020	29	NaN	NaN	8	4	11	7	2954010	290018812	...	NaN	NaN	NaN

157076 rows x 148 columns

```
[4] df.shape

    (157076, 148)
```

Figura 1 – Dados da PNAD – COVID - 1
Fonte: Elaborado pelo autor (2023)

Vejam que, logo de início, já temos a primeira complexidade. Há muitas colunas com códigos, são 148 características que foram analisadas pelo IBGE. Para a nossa sorte, o IBGE disponibiliza um dicionário que nos ajuda a compreender o que significa cada coluna.

Faremos uma primeira seleção das características do nosso dataset para facilitar o nosso entendimento e correlacionaremos com o dicionário do IBGE.

```
# Seleciona as características para o estudo.
df_nov2020 = pd.DataFrame(df,columns = ['UF', 'V1013', 'V1022', 'A002', 'A003', 'A004', 'B0011', 'B0012', 'B0014',
                                        'B00111','B007','B008', 'C005', 'C007', 'C003'])
```

```
df_nov2020.head()
```

	UF	V1013	V1022	A002	A003	A004	B0011	B0012	B0014	B00111	B007	B008	C005	C007	C003
0	11	11	1	36	1	4	2	2	2	2	1	2	NaN	4.0	NaN
1	11	11	1	30	2	4	2	2	2	2	2	2	NaN	7.0	NaN
2	11	11	1	13	1	4	2	2	2	2	1	2	NaN	NaN	NaN
3	11	11	1	11	1	4	2	2	2	2	1	2	NaN	NaN	NaN
4	11	11	1	57	2	1	2	2	2	2	2	2	NaN	NaN	NaN

```
df_nov2020.shape
```

```
(157076, 15)
```

Figura 2 – Dados da PNAD – COVID - 2
Fonte: Elaborado pelo autor (2023)

1. RENOMEAR O NOME DAS COLUNAS

A próxima etapa é renomear as colunas, pois imagine você preparando um modelo e sempre tendo que consultar o dicionário fornecido pelo o IBGE. Melhor alterarmos, correto?

```
# Renomear as colunas
df_nov2020 = df_nov2020.rename(columns = {'UF':'Estado',
                                          'V1013':'Mes',
                                          'V1022':'Area',
                                          'A002':'Idade_Morador',
                                          'A003':'Sexo',
                                          'A004':'Cor_Raca',
                                          'B0011':'Febre',
                                          'B0012':'Tosse',
                                          'B0014':'Dific_respiratoria',
                                          'B00111':'Cheiro_sabor',
                                          'B007':'Plano_Saude',
                                          'B008':'Teste_Covid',
                                          'C005':'Tempo_afastamento',
                                          'C007':'Carac_trabalho',
                                          'C003':'Motivo_afastamento'})
```

```
df_nov2020.head()
```

	Estado	Mes	Area	Idade_Morador	Sexo	Cor_Raca	Febre	Tosse	Dific_respiratoria	Cheiro_sabor	Plano_Saude	Teste_Covid	Tempo_afastame
0	11	11	1	36	1	4	2	2	2	2	1	2	N
1	11	11	1	30	2	4	2	2	2	2	2	2	N
2	11	11	1	13	1	4	2	2	2	2	1	2	N
3	11	11	1	11	1	4	2	2	2	2	1	2	N
4	11	11	1	57	2	1	2	2	2	2	2	2	N

Figura 3 – Dados da PNAD – COVID - 3
Fonte: Elaborado pelo autor (2023)

Agora que arrumamos uma base, vamos analisar os tipos das colunas para verificarmos se há a necessidade de trocar o tipo? Mas por que precisamos verificar esse tipo de informação?

A necessidade é deixar as características na formação em que elas devem ser. Concordam que, se o campo é Número Inteiro e na nossa base estiver como Texto, não será possível fazer quaisquer cálculos? Então, essa é uma primeira pratica de feature engineering a ser feita. Para fazer essa conversão de tipo (Dtype), existem várias funções que são nativas do Python, sendo as mais utilizadas:

- .to_string;
- .astype(float);
- .to_numeric

```
df_nov2020.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157076 entries, 0 to 157075
Data columns (total 15 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   Estado               157076 non-null  int64  
 1   Mes                  157076 non-null  int64  
 2   Area                 157076 non-null  int64  
 3   Idade_Morador        157076 non-null  int64  
 4   Sexo                 157076 non-null  int64  
 5   Cor_Raca             157076 non-null  int64  
 6   Febre                157076 non-null  int64  
 7   Tosse                157076 non-null  int64  
 8   Dific_respiratoria   157076 non-null  int64  
 9   Cheiro_sabor         157076 non-null  int64  
 10  Plano_Saude          157076 non-null  int64  
 11  Teste_Covid          157076 non-null  int64  
 12  Tempo_afastamento    2327 non-null    float64
 13  Carac_trabalho       53820 non-null   float64
 14  Motivo_afastamento   4892 non-null    float64
dtypes: float64(3), int64(12)
memory usage: 18.0 MB
```

Figura 4 – Informações do PNAD
Fonte: Elaborado pelo autor (2023)

No nosso caso, não precisamos fazer a conversão, mas é super importante que os Dtype sejam verificados.

2. DADOS AUSENTES

Não tem jeito, temos que saber lidar com dados ausentes. A maioria dos modelos de ML não funcionam se houver dados ausentes, mas não há uma “receita de bolo” que ensine a como lidar com esses dados.

A pergunta sempre fica, “será que esses dados ausentes farão falta na nossa base de dados?”. Ainda mais com dados que vem de pesquisas, como é no nosso caso, será que elas não querem representar alguma coisa? É sempre complicado, mas há diversas formas de lidar com esses dados:

- Remover as linhas que tenham dados ausentes;
- Remover as colunas que tenham dados ausentes;
- Imputar dados aos valores ausentes;
- Criar uma coluna dizendo que tal característica teria dados ausentes.

Analisando o nosso conjunto de dados, é possível visualizar que temos 3 colunas com dados ausentes.

```
df_nov2020.isnull().sum()
```

```
Estado                   0
Mes                      0
Area                     0
Idade_Morador            0
Sexo                     0
Cor_Raca                 0
Febre                    0
Tosse                    0
Dific_respiratoria       0
Cheiro_sabor             0
Plano_Saude              0
Teste_Covid              0
Tempo_afastamento     154749
Carac_trabalho        103256
Motivo_afastamento    152184
dtype: int64
```

Figura 5 – Dados ausentes.
Fonte: Elaborado pelo autor (2023)

Para descartamos os dados ausentes, existem 3 formas:

- Deletar qualquer linha que tenha uma informação nula;
    - .dropna()

```
df_nov2020.dropna()
```

	Estado	Mes	Area	Idade_Morador	Sexo	Cor_Raca	Febre	Tosse	Dific_respiratoria	Cheiro_sabor	Plano_Saude	Teste_Covid	Tempo_afas
316	11	11	1	61	2	1	2	2	2	2	1	1	
432	11	11	1	41	2	4	2	2	2	2	2	1	
519	11	11	1	33	1	4	2	2	2	2	2	2	
952	11	11	1	65	1	1	2	2	2	2	2	2	
991	11	11	1	38	2	1	2	2	2	2	2	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	
156451	29	11	2	41	2	2	2	2	2	2	2	2	
156492	29	11	2	18	1	4	2	2	2	2	2	2	
156613	29	11	2	37	1	4	2	2	2	2	2	2	
156669	29	11	2	29	2	4	2	2	2	2	2	2	
156845	29	11	2	28	2	4	2	2	2	2	2	2	

767 rows × 15 columns

Figura 6 – .dropna()
Fonte: Elaborado pelo autor (2023)

- Deletar a coluna que tiver os dados nulos e não será utilizado;
    - .dropna(columns = “nome_coluna”)

```
df_nov2020.drop(columns = 'Tempo_afastamento', inplace = True)
df_nov2020.drop(columns = 'Carac_trabalho', inplace = True)
df_nov2020.drop(columns = 'Motivo_afastamento', inplace = True)
```

```
df_nov2020
```

	Estado	Mes	Area	Idade_Morador	Sexo	Cor_Raca	Febre	Tosse	Dific_respiratoria	Cheiro_sabor	Plano_Saude	Teste_Covid
0	11	11	1	36	1	4	2	2	2	2	1	2
1	11	11	1	30	2	4	2	2	2	2	2	2
2	11	11	1	13	1	4	2	2	2	2	1	2
3	11	11	1	11	1	4	2	2	2	2	1	2
4	11	11	1	57	2	1	2	2	2	2	2	2
...	...	...	...	...	...	...	...	...	...	...	...	...
381433	53	11	2	45	2	4	2	2	2	2	2	2
381434	53	11	2	22	2	4	2	2	2	2	2	2
381435	53	11	2	16	2	4	2	2	2	2	2	2
381436	53	11	2	83	1	1	2	2	2	2	2	2
381437	53	11	2	75	2	4	2	2	2	2	2	2

381438 rows × 12 columns

Figura 7 – .dropna(columns = “nome_coluna”)
Fonte: Elaborado pelo autor (2023)

- Deletar a coluna que tiverem dados ausentes.
    - .dropna (axis = 13)

```
df_nov2020.dropna(axis = 1)
```

	Estado	Mes	Area	Idade_Morador	Sexo	Cor_Raca	Febre	Tosse	Dific_respiratoria	Cheiro_sabor	Plano_Saude	Teste_Covid
0	11	11	1	36	1	4	2	2	2	2	1	2
1	11	11	1	30	2	4	2	2	2	2	2	2
2	11	11	1	13	1	4	2	2	2	2	1	2
3	11	11	1	11	1	4	2	2	2	2	1	2
4	11	11	1	57	2	1	2	2	2	2	2	2
...	...	...	...	...	...	...	...	...	...	...	...	...
381433	53	11	2	45	2	4	2	2	2	2	2	2
381434	53	11	2	22	2	4	2	2	2	2	2	2
381435	53	11	2	16	2	4	2	2	2	2	2	2
381436	53	11	2	83	1	1	2	2	2	2	2	2
381437	53	11	2	75	2	4	2	2	2	2	2	2

381438 rows × 12 columns

Figura 8 – .dropna(axis = 13)
Fonte: Elaborado pelo autor (2023)

Normalização e padronização dos dados

A normalização e padronização são técnicas frequentemente aplicadas na etapa de preparar os dados para construção do nosso conjunto de dados, para que os modelos de ML funcionem de forma correta, pois os dados são colocados em intervalos comuns.

Não são técnicas obrigatórias, já que há modelos que não dependem dessa padronização, pois os algoritmos entendem que não normalizar também funciona. Um exemplo de modelo de ML que não precisa normalizar é o Decision Tree. No entanto, há algoritmos que necessitam a normalização, como por exemplo, o SVM.

A normalização coloca os dados de intervalos entre 0 e 1 ou -1 e 1, caso haja valores negativos, sem se preocupar com as diferentes faixas de valores, e isso faz com que os outliers sejam englobados no conjunto de dados sem que os mesmos sejam excluídos.

Já a padronização tem basicamente o mesmo intuito da normalização, colocar os dados em uma mesma escala. Porém, a padronização coloca os dados entre 0 e ao desvio padrão. Esse algoritmo é melhor utilizado quando há uma distribuição Gaussiana (Distribuição Normal).

Os algoritmos de construção de ML que necessitam de normalização são, basicamente: KNN (K-Nearest Neighbours), Redes Neurais, Regressão Linear, Regressão Logística e SVM.

Os algoritmos de construção de ML que não necessitam de normalização são: Árvores de Decisão, Random Forest, AdaBoost e Naïve Bayes.

Mas vale ressaltar, mesmo não precisando de normalização, é sempre bom testar uma das técnicas.

Que tal dar uma lida em um artigo de Feature Engineering no blog do alteryx? Ele traz uma boa visão sobre o assunto:

Artigo do Alteryx.

O QUE VOCÊ VIU NESTA AULA?

Introdução à Feature Engineering; Limpeza de dados; Padronização de dados; Normalização de dados.

Daqui em diante, é importante que você replique os conhecimentos adquiridos para fortalecer mais suas bases e conhecimentos.

IMPORTANTE: não esqueça de praticar com o desafio da disciplina, para que assim você possa aprimorar os seus conhecimentos!

Você não está só nesta jornada! Te esperamos no Discord e nas *lives* com os professores(as) especialistas, onde você poderá tirar dúvidas, compartilhar conhecimentos e estabelecer conexões!

REFERÊNCIAS

DOCUMENTAÇÃO PANDAS. **Disponível em: <https://pandas.pydata.org/>.** Acesso em: 04 mai 2023.

GOOGLE COLAB. **Disponível em: <https://colab.research.google.com/>.** Acesso em: 04 mai 2023.

GRUS, Joel. **Data Science do Zero**. Rio de Janeiro: Alta Books Editora, 2016.

HARRISON, Matt. **Machine Learning**: guia de referência rápida - trabalhando com dados estruturados em python. São Paulo: O'Reilly Media, 2019.

IBGE. Disponível em: **<**https://www.ibge.gov.br/estatisticas/sociais/trabalho/27946-divulgacao-semanal-pnadcovid1.html?=&t=downloads**>.** Acesso em: 04 mai 2023.

PALAVRAS-CHAVE

Python. Pandas. Feature Engeneering. Limpeza de dados. Normalização de dados. Padronização de dados.