

Data Science

FEUP

Ricardo Pinto



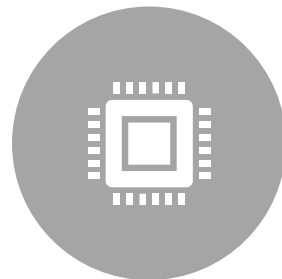


0,65	0,00	%	SOJK
0,44	0,82	%	LBANK
0,04	0,00	%	TEG
0,49	0,22	%	TEK
0,62	0,62	%	TEENERG
0,08	0,00	%	TEXT
0,00	0,00	%	PROFK
0,78	0,26	%	TELET
0,71	0,21	%	TIK
0,22	0,22	%	TIKE
0,00	0,00	%	TIIP
0,12	0,00	%	TRAST
0,00	0,00	%	TROP
0,02	0,02	%	
0,02	0,02	%	
0,08	0,08	%	
0,00	0,00	%	
0,21	0,21	%	
0,00	0,00	%	

Ciência de Dados



recolha



processamento



análise



ferramentas &
linguagens



insights válidos



ética



Data Science e Machine Learning em Python



NumPy: manipulação de arrays e matrizes numéricas



Pandas: estrutura de dados e análise



Matplotlib: visualização de dados



Scikit-learn: aprendizado de máquina



TensorFlow: deep learning e redes neurais

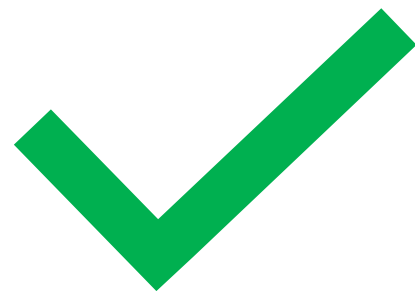


PyTorch: deep learning e redes neurais




VAMOS APRENDER TUDO! JÁ!





Emissions
Detection And
Reporting,
commonly
known as EDAR



1.Dados >  tempestades.csv


	data_evento	regiao	velocidade_vento	precipitacao
1	2020-01-06	Região A	80	150
2	2020-01-22	Região B	120	250
3	2020-02-12	Região C	60	100
4	2020-03-16	Região A	90	180
5	2020-04-01	Região B	110	220
6	2020-04-14	Região A	70	130
7	2020-05-26	Região C	50	90
8	2020-06-19	Região B	130	280
9	2020-07-06	Região A	85	155
10	2020-08-23	Região C	65	120

1.Dados >  incendios.csv

	data_evento	regiao	area_afetada	causa
1	2020-01-05	Região A	1200	Relâmpago
2	2020-01-21	Região B	2600	Atividade humana
3	2020-02-11	Região C	800	Relâmpago
4	2020-03-15	Região A	1500	Atividade humana
5	2020-03-30	Região B	2400	Relâmpago
6	2020-04-13	Região A	1300	Atividade humana
7	2020-05-25	Região C	900	Relâmpago
8	2020-06-18	Região B	2900	Atividade humana
9	2020-07-05	Região A	1100	Relâmpago
10	2020-08-22	Região C	700	Atividade humana

Proteção Civil

- tempestades
- terremotos
- incêndios
- inundações

1.Dados >  terremotos.csv

	data_evento	regiao	magnitude	profundidade
1	2020-01-04	Região A	3.5	10.0
2	2020-01-20	Região B	5.2	15.0
3	2020-02-10	Região C	4.1	8.0
4	2020-03-14	Região A	3.8	11.0
5	2020-03-29	Região B	5.6	16.0
6	2020-04-12	Região A	3.9	9.5
7	2020-05-24	Região C	4.3	7.0
8	2020-06-17	Região B	5.7	14.5
9	2020-07-04	Região A	3.6	12.0
10	2020-08-21	Região C	4.0	10.5

1.Dados >  inundacoes.csv

	data_evento	regiao	magnitude	duracao
1	2020-01-03	Região A	5.0	24
2	2020-01-15	Região B	6.2	48
3	2020-02-07	Região C	4.5	18
4	2020-03-11	Região A	5.5	36
5	2020-03-27	Região B	6.0	42
6	2020-04-10	Região A	5.3	30
7	2020-05-22	Região C	4.7	21
8	2020-06-15	Região B	6.4	54
9	2020-07-02	Região A	5.1	28
10	2020-08-19	Região C	4.9	16



```
import pandas as pd
```

```
# 1.1. Carregar os dados
```

```
# Substituir os nomes dos arquivos pelos arquivos de dados reais obtidos das fontes mencionadas.
```

```
inundacoes2020 = pd.read_csv("inundacoes_a.csv")
```

```
inundacoes2021 = pd.read_csv("inundacoes_a2.csv")
```

```
inundacoes2022 = pd.read_csv("tudo_menos_inundacoes.csv")
```

```
# 1.2. Unificar os conjuntos de dados
```

```
# Concatenar todos os conjuntos de dados num único DataFrame.
```

```
data = pd.concat([inundacoes2020, inundacoes2021, inundacoes2022], ignore_index=True)
```

```
# Adicionar uma coluna 'tipo_desastre' para cada conjunto de dados para identificar o tipo de desastre.
```

```
data["tipo_desastre"] = "inundacao"
```



```
# 1.3. Limpar os dados
# Remover linhas sem dados
data.dropna(inplace=True)

# Corrigir inconsistências, se houver
# Exemplo: converter strings para lowercase
data["regiao"] = data["regiao"].str.lower()

# 1.4. Conversão de tipos de dados
# Converter colunas de data e hora para o tipo datetime
data["data_evento"] = pd.to_datetime(data["data_evento"])

# Converter colunas numéricas para o tipo apropriado (float, int)
data["magnitudo"] = data["magnitudo"].astype(float)
data["duracao"] = data["duracao"].astype(int)

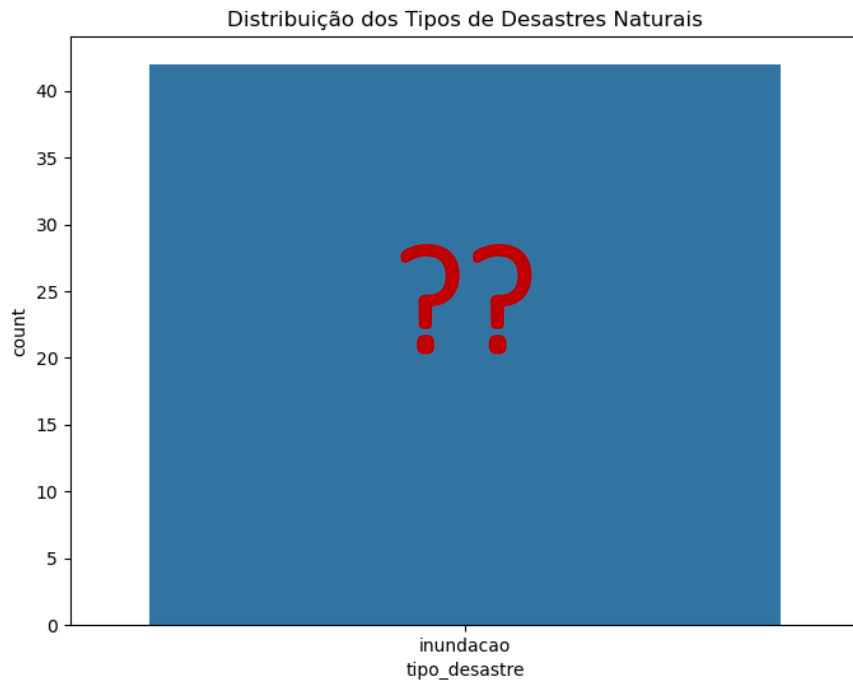
# Verificar se os dados foram carregados e preparados corretamente
print(data.head())
```

```
(base) D:\Dropbox\0.Transferencia\2.Courses\Workshop FEUP_DataScience\slides-code\etapa1>python etapa1.py
  data_evento  regiao  magnitudo  duracao  tipo_desastre
0  2020-01-03  região a        5.0       24    inundacao
1  2020-01-15  região b        6.2       48    inundacao
2  2020-02-07  região c        4.5       18    inundacao
3  2020-03-11  região a        5.5       36    inundacao
4  2020-03-27  região b        6.0       42    inundacao
```

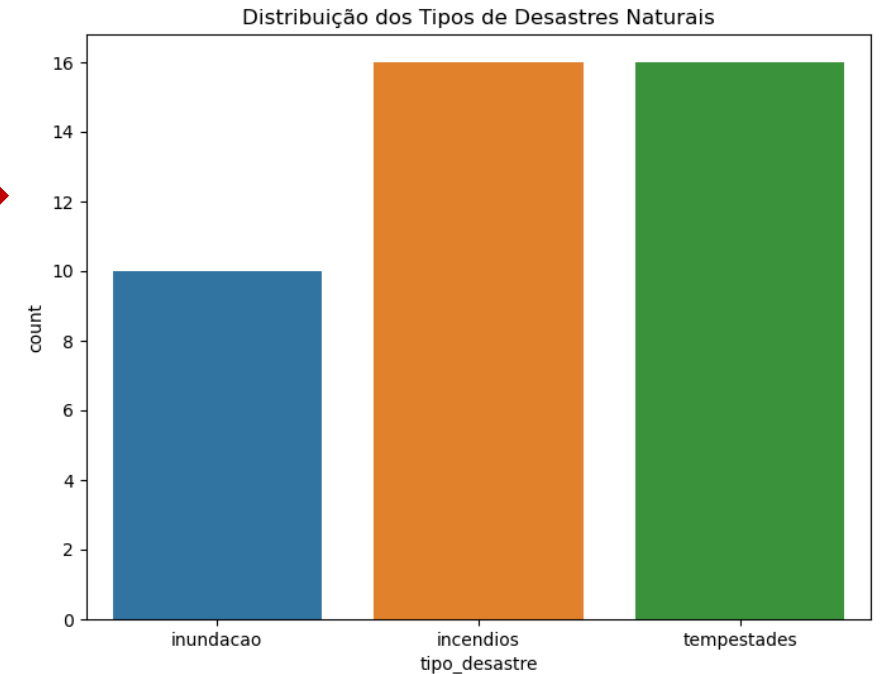
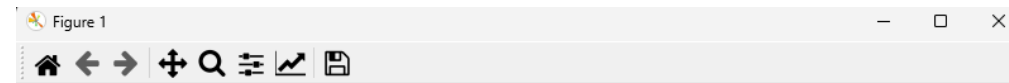
```
import matplotlib.pyplot as plt
import seaborn as sns
```

2.1. Visualizar a distribuição dos desastres naturais

```
plt.figure(figsize=(8, 6))
sns.countplot(x='tipo_desastre', data=data)
plt.title('Distribuição dos Tipos de Desastres Naturais')
plt.show()
```



Queria isto! 😞



```
# 2.2. Visualizar a distribuição dos desastres naturais ao longo do tempo
```

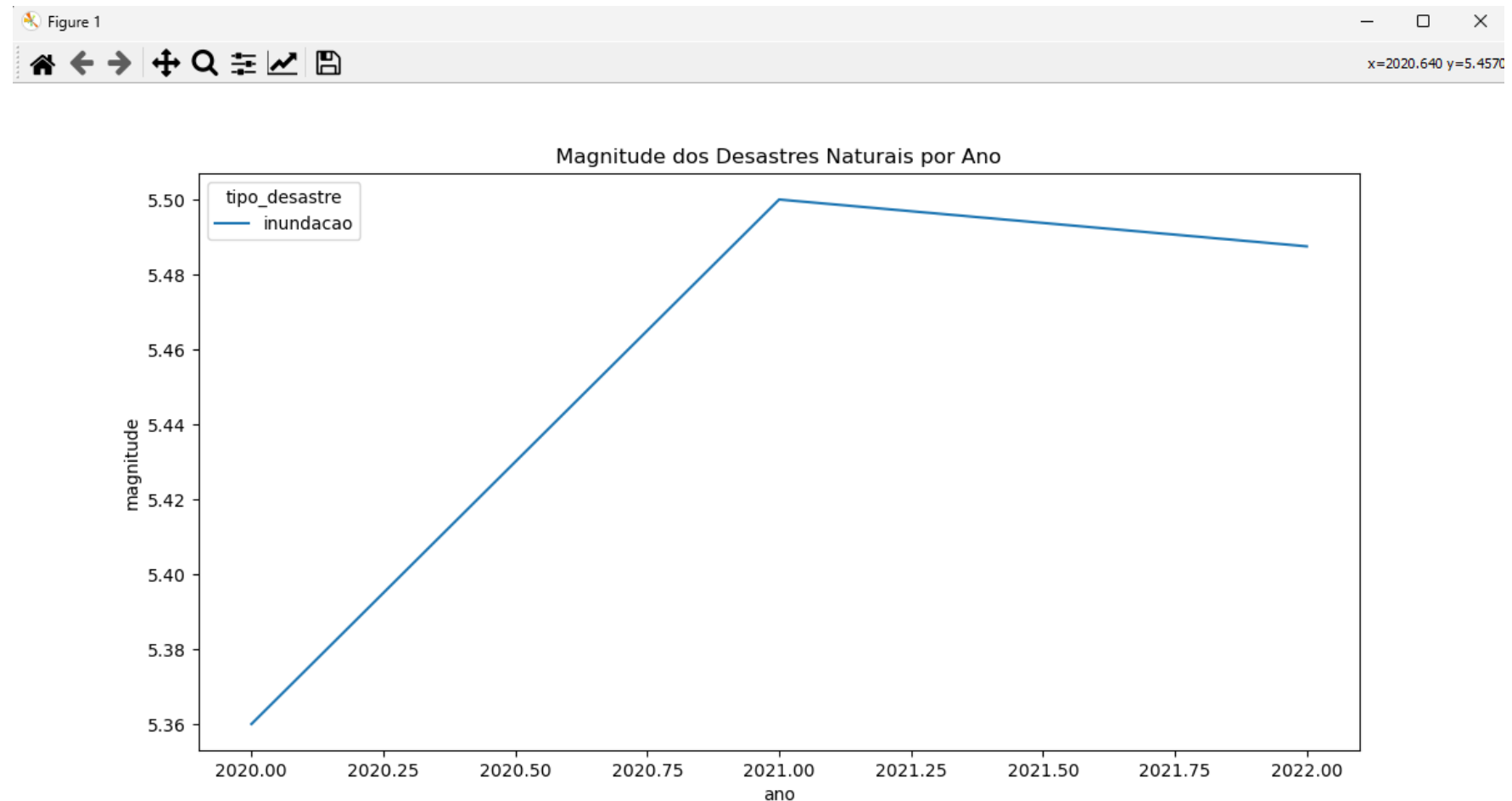
```
data['ano'] = data['data_evento'].dt.year
```

```
plt.figure(figsize=(12, 6))
```

```
sns.lineplot(x='ano', y='magnitude', hue='tipo_desastre', data=data, ci=None)
```

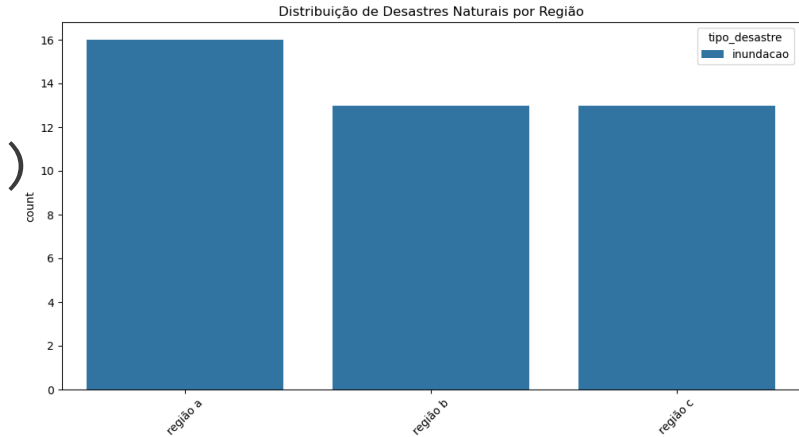
```
plt.title('Magnitude dos Desastres Naturais por Ano')
```

```
plt.show()
```



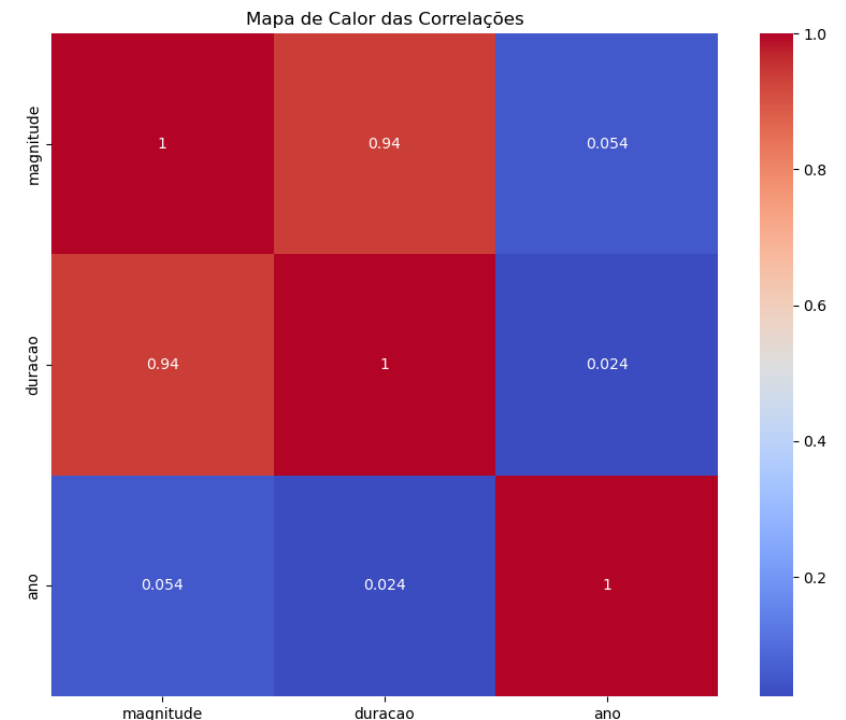
2.3. Visualizar a distribuição de desastres por região

```
plt.figure(figsize=(12, 6))  
sns.countplot(x='regiao', hue='tipo_desastre', data=data)  
plt.title('Distribuição de Desastres Naturais por Região')  
plt.xticks(rotation=45)  
plt.show()
```



2.4. Visualizar correlações entre variáveis

```
plt.figure(figsize=(10, 8))  
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')  
plt.title('Mapa de Calor das Correlações')  
plt.show()
```



2.5. Análise adicional

Explorar outras visualizações e padrões conforme necessário.


```

# 3.1. Criar novas variáveis
# Média histórica de eventos por mês
data['mes'] = data['data_evento'].dt.month
eventos_por_mes = data.groupby(['mes', 'tipo_desastre']).size().unstack(fill_value=0)
data['media_eventos_mes'] = data.apply(lambda x: eventos_por_mes.loc[x['mes'],
x['tipo_desastre']], axis=1)

# Média de eventos por região
eventos_por_regiao = data.groupby(['regiao', 'tipo_desastre']).size().unstack(fill_value=0)
data['media_eventos_regiao'] = data.apply(lambda x: eventos_por_regiao.loc[x['regiao'],
x['tipo_desastre']], axis=1)

# 3.2. Normalizar as variáveis numéricas
# Selecionar colunas numéricas (excluindo colunas categóricas e de data)
numeric_columns = ['magnitudo', 'duracao', 'media_eventos_mes', 'media_eventos_regiao']

# Usar MinMaxScaler para normalizar os dados
scaler = MinMaxScaler()
data[numeric_columns] = scaler.fit_transform(data[numeric_columns])

# Verificar se os dados foram
# processados corretamente
print(data.head())

```

```

(base) D:\Dropbox\0.Transferencia\2.Courses\Workshop FEUP_DataScience\slides-code\etapa3>python etapa3.py

```

	data_evento	regiao	magnitudo	duracao	tipo_desastre	mes	media_eventos_mes	media_eventos_regiao
0	2020-01-03	região a	0.357143	0.238095	inundacao	1	1.00	1.0
1	2020-01-15	região b	0.785714	0.809524	inundacao	1	1.00	0.0
2	2020-02-07	região c	0.178571	0.095238	inundacao	2	0.75	0.0
3	2020-03-11	região a	0.535714	0.523810	inundacao	3	0.75	1.0
4	2020-03-27	região b	0.714286	0.666667	inundacao	3	0.75	0.0

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```
# 4.1. Preparar os dados para treino e teste
```

```
# Selecionar as colunas de recursos (X) e a coluna alvo (y)
```

```
X = data.drop(['data_evento', 'tipo_desastre', 'regiao'], axis=1)
```

```
y = data['tipo_desastre']
```

```
# Dividir o conjunto de dados em treino e teste
```

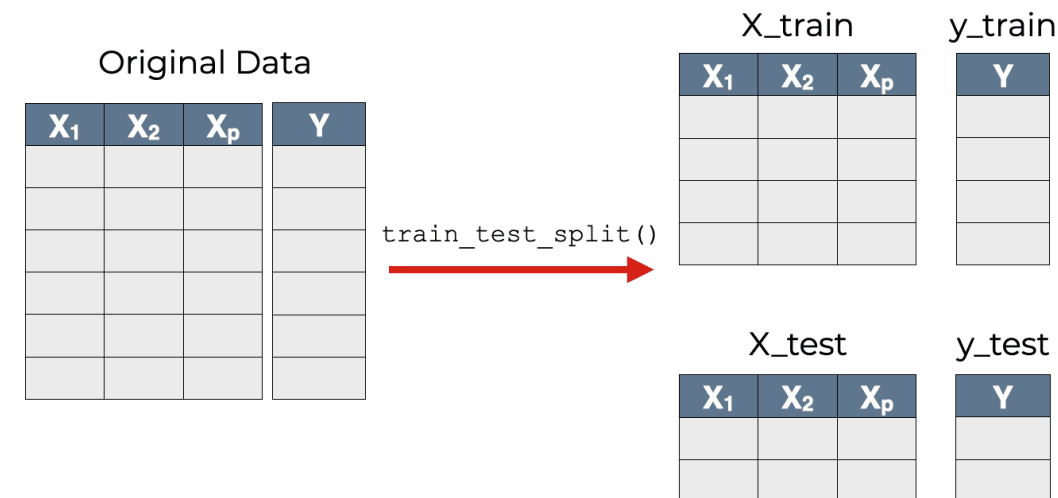
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

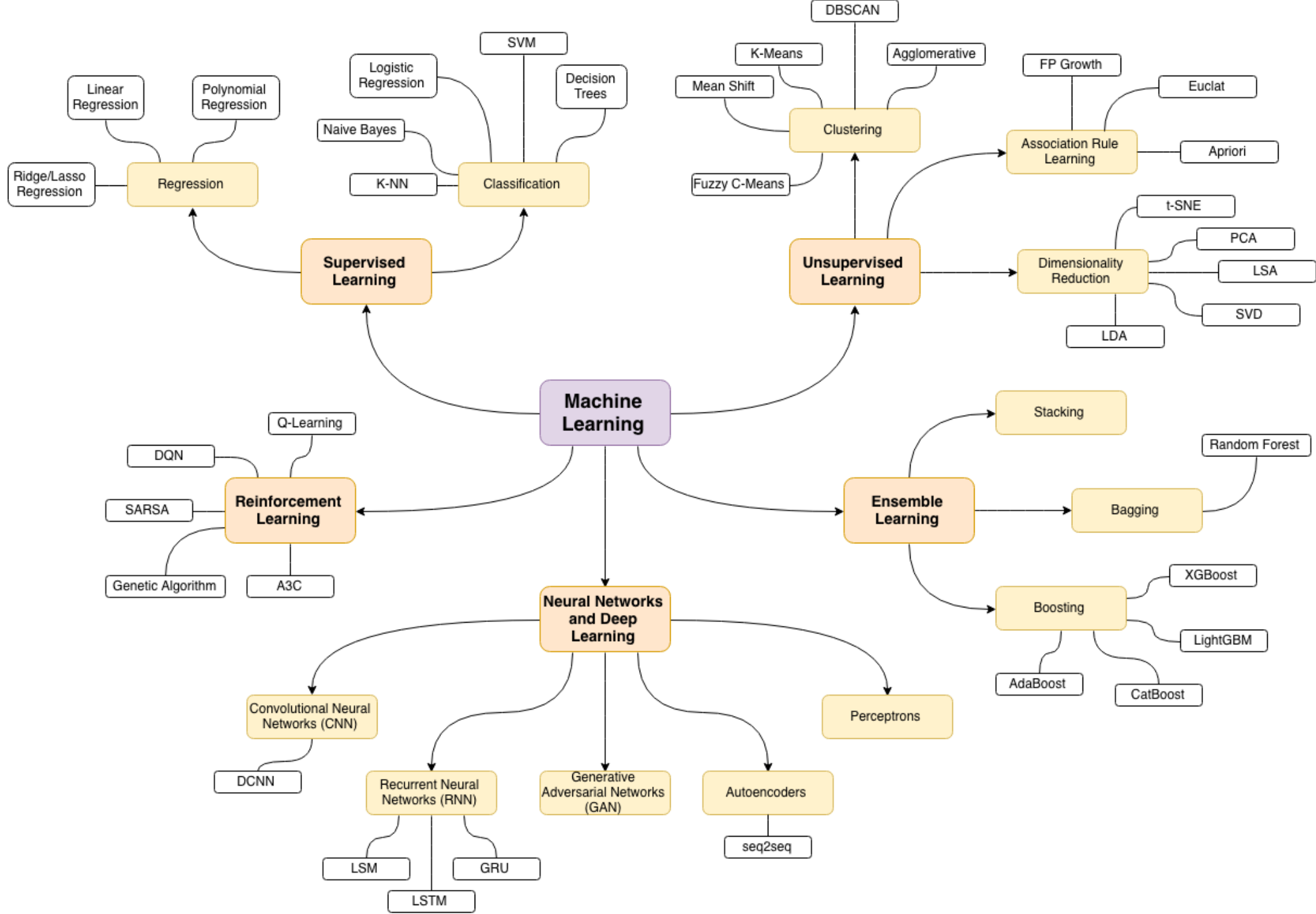
```
# 4.2. Escolher e treinar o modelo de Machine Learning
```

```
# Neste exemplo, usamos o RandomForestClassifier, mas pode-se experimentar outros modelos,
como XGBoost ou Redes Neurais
```

```
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
model.fit(X_train, y_train)
```

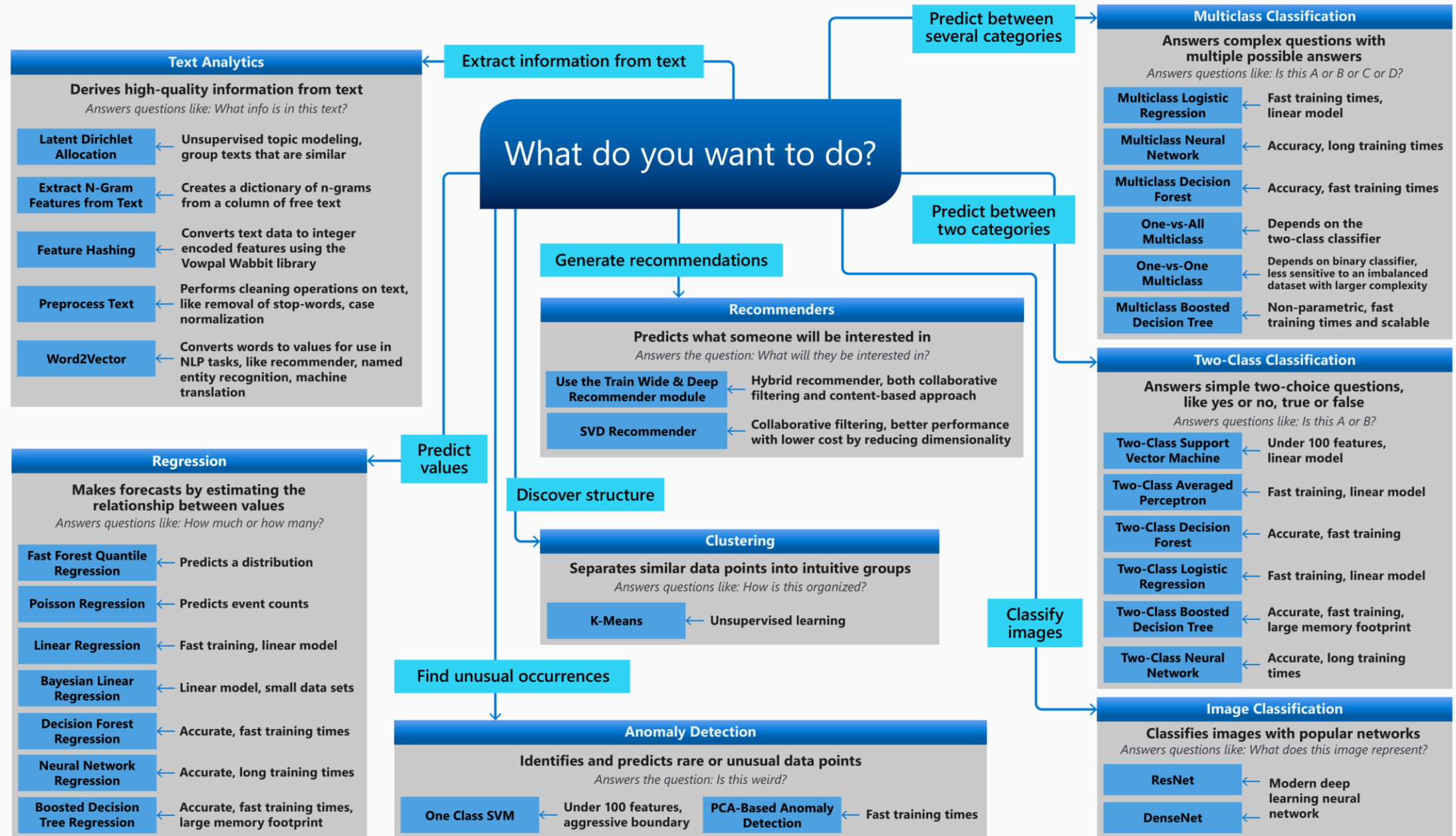






Machine Learning Algorithm Cheat Sheet

This cheat sheet helps you choose the best machine learning algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the goal you want to achieve with your data.




```
from sklearn.metrics import classification_report, accuracy_score
```

```
# 5.1. Testar o modelo usando os dados de teste
```

```
y_pred = model.predict(X_test)
```

```
# 5.2. Avaliar a performance do modelo
```

```
# Relatório de classificação
```

```
print("Relatório de classificação:\n", classification_report(y_test, y_pred))
```

```
# Precisão ou Exatidão?
```

```
print("Precisão ou Exatidão?", accuracy_score(y_test, y_pred))
```

```
# Em machine learnig
```

```
# Precisão - consistência ou reprodutibilidade dos resultados
```

```
# Exatidão - ou acurácia - quão próximos os resultados estão dos valores verdadeiros ou reais.
```

```
(base) D:\Dropbox\0.Transferencia\2.Courses\Workshop FEUP_DataScience\slides-code\etapa5>python etapa5.py
Relatório de classificação:
              precision    recall  f1-score   support

   inundacao         1.00      1.00      1.00        13

   accuracy                   1.00         13
  macro avg         1.00      1.00      1.00         13
 weighted avg         1.00      1.00      1.00         13

Exatidão: 1.0
```

```
# 5.3. Análise adicional
```

```
# Se necessário, explorar outras métricas, matriz de confusão, curvas ROC e PR, e
```

```
# ajuste os parametros do modelo
```

ERROS? PADRÕES!

```
import pickle

# 6.1. Exportar o modelo treinado
with open('modelo_desastres_naturais.pkl', 'wb') as file:
    pickle.dump(model, file)

# 6.2. Importar o modelo treinado em um ambiente de produção
with open('modelo_desastres_naturais.pkl', 'rb') as file:
    modelo_producao = pickle.load(file)

# 6.3. Fazer previsões em tempo real
# Exemplo de dados de entrada para prever o tipo de desastre natural
entrada_exemplo = {
    'magnitude': 5.5, 'duracao': 10, 'mes': 6, 'media_eventos_mes': 3,
    'media_eventos_regiao': 5
}

# Converter o dicionário de entrada em um DataFrame
entrada_df = pd.DataFrame([entrada_exemplo])

# Normalizar a entrada usando o mesmo scaler utilizado no treino
entrada_df[numeric_columns] = scaler.transform(entrada_df[numeric_columns])

# Fazer a previsão usando o modelo importado
previsao = modelo_producao.predict(entrada_df)
print("Previsão do tipo de desastre natural:", previsao[0])
```

```
(base) D:\Dropbox\0.Transferencia\2.Courses\Workshop FEUP_DataScience\
Previsão do tipo de desastre natural: inundacao
```




Agora a analisar bem!

“A chuva que caiu com força, a partir das 15h00, na região provocou diversas inundações, derrocadas e quedas de árvores em vários concelhos do distrito de Viseu.

Na cidade de Viseu, o **rio Pavia** voltou a galgar as margens, nomeadamente, junto ao recinto da feira semanal...”

<https://www.diarioviseu.pt/noticia/90843>

Vamos fazer um modelo de predição de inundação no Pavia!