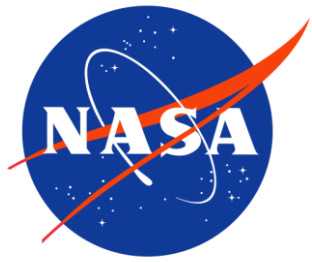
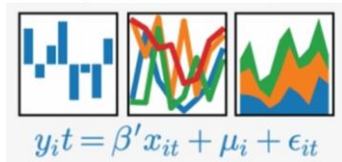
@Odemir Depieri Jr

Guia de Machine Learning - Floresta de Decisão Python

Case com os dados da NASA



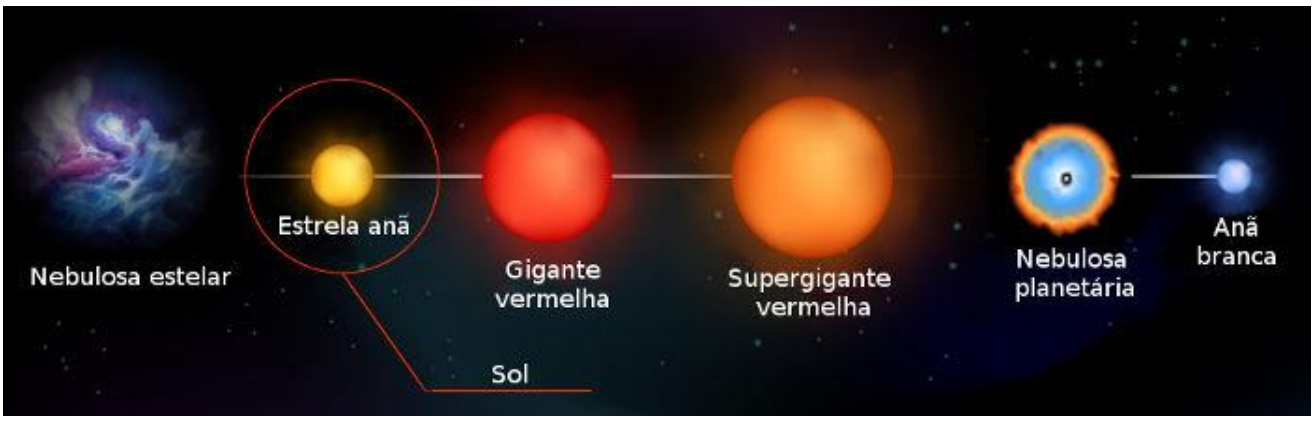


@Odemir Depieri Jr

Resumo sobre Dados da NASA

Esse banco de dados é sobre Classificação de Estrelas.

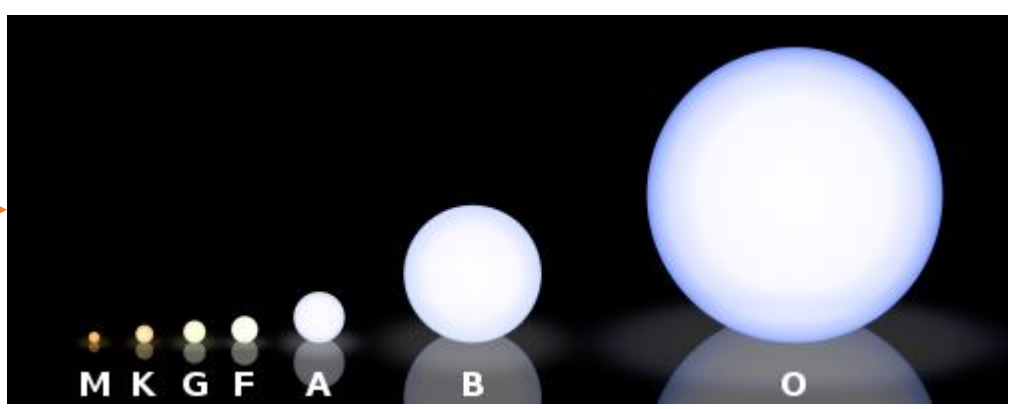
1º Vamos entender os tipo de estrelas que existem no Universo.



Na imagem há alguns exemplos de estrelas que existem pelo cosmo apenas para ilustrar. **Obs:** Existem outros tipos.

2º Vamos entender um pouco sobre Física (tipos espectrais de luz) que as estrelas emitem.

Classe	Temperatura[1] (kelvin)	Cor convencional	Cor aparente[2][3]	Massa[1] (massas solares)	Raio[1] (raios solares)	Luminosidade bolométrica[1] ($L_\odot$)	Linhas de hidrogênio	Fração das estrelas da sequência principal[4]
O	≥ 33 000 K	azul	azul	≥ 16 $M_\odot$	≥ 6,6 $R_\odot$	≥ 30 000 $L_\odot$	Fracas	~0,00003%
B	10 000–33 000 K	azul-branco	azul-branco	2,1–16 $M_\odot$	1,8–6,6 $R_\odot$	25–30 000 $L_\odot$	Moderadas	0,13%
A	7 500–10 000 K	branco	branco a azul-branco	1,4–2,1 $M_\odot$	1,4–1,8 $R_\odot$	5–25 $L_\odot$	Fortes	0,6%
F	6 000–7 500 K	amarelo-branco	branco	1,04–1,4 $M_\odot$	1,15–1,4 $R_\odot$	1,5–5 $L_\odot$	Moderadas	3%
G	5 200–6 000 K	amarelo	amarelo-branco	0,8–1,04 $M_\odot$	0,96–1,15 $R_\odot$	0,6–1,5 $L_\odot$	Fracas	7,6%
K	3 700–5,200 K	laranja	amarelo-laranja	0,45–0,8 $M_\odot$	0,7–0,96 $R_\odot$	0,08–0,6 $L_\odot$	Muito fracas	12,1%
M	2 000–3 700 K	vermelho	laranja-vermelho	≤ 0,45 $M_\odot$	≤ 0,7 $R_\odot$	≤ 0,08 $L_\odot$	Muito fracas	76,45%



As classes indicam a temperatura da atmosfera da estrela e são normalmente listadas da mais quente para a mais fria.

Resumo sobre Dados da NASA

3º Vamos entender que tipo de informação há nos dados. Foi disponibilizado uma breve descrição dos dados no enunciado.

Descrição

Classificação do tipo estrela

Para comparar todos os modelos de ML,
pode ser usado para previsão

Temperatura - K
L - L / Lo
R - R / Ro
AM -
Cor Mv - Cor Geral do Espectro
Spectral_Class - O, B, A, F, G, K, M / SMASS - https: // en.wikipedia.org/wiki/Asteroid_spectral_types
Type - Red Dwarf, Brown Dwarf, White Dwarf, Main Sequence, Super Giants, Hyper Giants

ALVO:
Tipo

de 0 a 5

- Anã Vermelha - 0
- Anã Marrom - 1
- Anã Branca - 2
- Sequência Principal - 3
- Super Gigantes - 4
- Hiper Gigantes - 5

MATEMÁTICA:

Lo = 3,828 × 10 ^ 26 Watts
(luminosidade média do sol)
Ro = 6,9551 × 10 ^ 8 m
(raio médio do sol)

^

Link para acessar os dados.

https://www.kaggle.com/brsdincer/star-type-classification

Mão na Massa !

Vamos explorar os dados

```
[2]  # Biblioteca para modelagem de dados
     import pandas as pd

     # Biblioteca para recursos matemáticos
     import numpy as np

     # Bibliotecas de plotagem de dados
     import seaborn as sns
     import matplotlib.pyplot as plt

     # Biblioteca/Função para ignorar avisos
     from warnings import filterwarnings
```

```
[4]  # Lendo a Base de Dados
     Base_Dados = pd.read_csv('Stars.csv')
```

```
[6]  # Verificando as primeiras linhas
     Base_Dados.head()
```

	Temperature	L	R	A_M	Color	Spectral_Class	Type
0	3068	0.002400	0.1700	16.12	Red	M	0
1	3042	0.000500	0.1542	16.60	Red	M	0
2	2600	0.000300	0.1020	18.70	Red	M	0
3	2800	0.000200	0.1600	16.65	Red	M	0
4	1939	0.000138	0.1030	20.06	Red	M	0

```
[82] # Verificando as colunas
     for c in Base_Dados.columns:
       print(c)
```

```
Temperature
L
R
A_M
Color
Spectral_Class
Type
```

```
[17] # Verificando a dimensão da base de dados
     Base_Dados.shape
```

```
(240, 7)
```

Mão na Massa !

```
[10] # Verificando o formato dos campos
     Base_Dados.info()
```

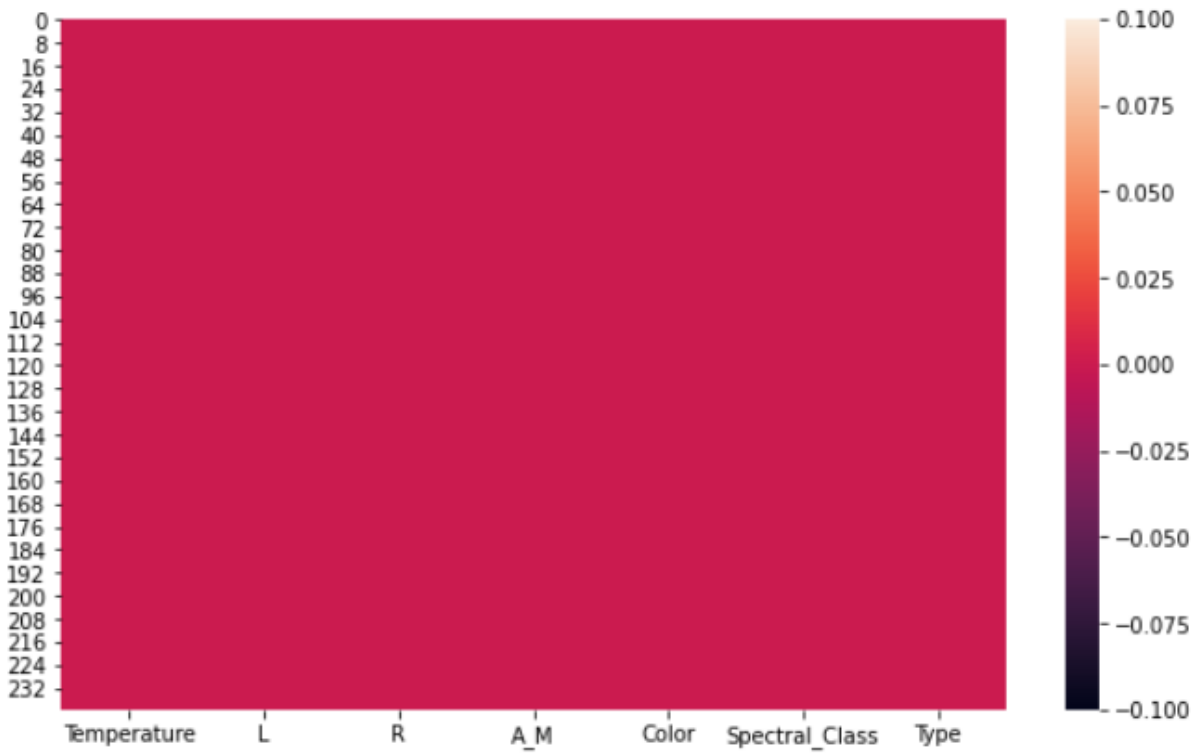
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Temperature     240 non-null    int64
 1   L               240 non-null    float64
 2   R               240 non-null    float64
 3   A_M             240 non-null    float64
 4   Color           240 non-null    object
 5   Spectral_Class  240 non-null    object
 6   Type            240 non-null    int64
dtypes: float64(3), int64(2), object(2)
memory usage: 13.2+ KB
```

```
[8] # Gerando algumas estatística para entender um pouco os dados

    # Dicionário para entender as estatísticas abaixo:
    # count --> Total de registros
    # mean --> Média
    # std --> Desvio Padrão
    # min --> Valor mínimo
    # 25% --> 1º Quartil
    # 50% --> Mediana
    # 75% --> 3º Quartil
    # max --> Valor Maior

    # Comando para gerar estatísticas sobre os dados
    Base_Dados.describe()
```

	Temperature	L	R	A_M	Type
count	240.000000	240.000000	240.000000	240.000000	240.000000
mean	10497.462500	107188.361635	237.157781	4.382396	2.500000
std	9552.425037	179432.244940	517.155763	10.532512	1.711394
min	1939.000000	0.000080	0.008400	-11.920000	0.000000
25%	3344.250000	0.000865	0.102750	-6.232500	1.000000
50%	5776.000000	0.070500	0.762500	8.313000	2.500000
75%	15055.500000	198050.000000	42.750000	13.697500	4.000000
max	40000.000000	849420.000000	1948.500000	20.060000	5.000000

Mão na Massa !

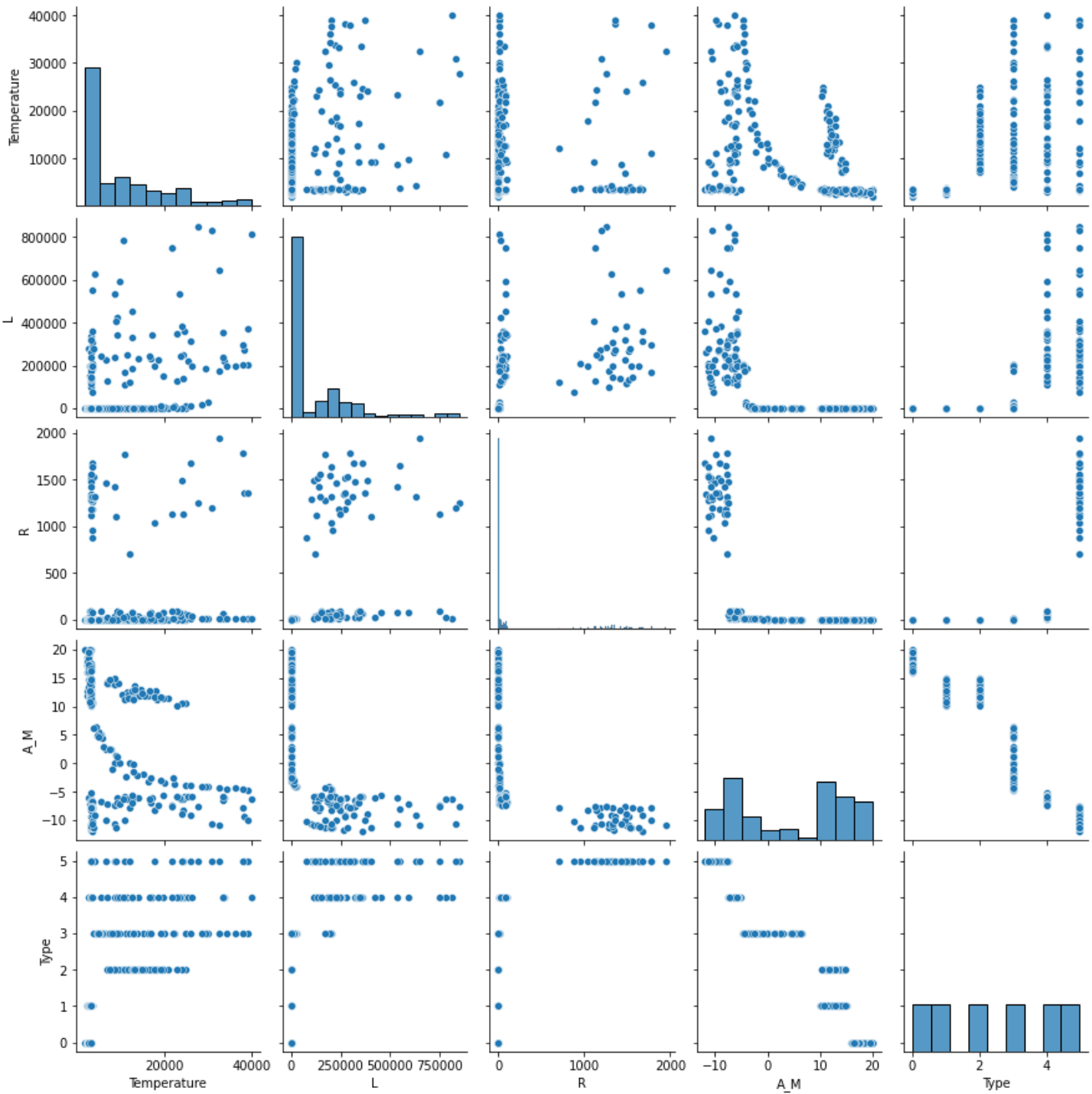
```
# Verificando se há um valor nulo na base de dados
# Caso exista haverá linhas brancas no gráfico
plt.figure( figsize=(10,6) )
sns.heatmap( Base_Dados.isnull() );
```




```
# Verificando se há alguma correlação entre as variaveis
plt.figure( figsize=(10,6) )
sns.heatmap( Base_Dados.corr(), annot=True );
```

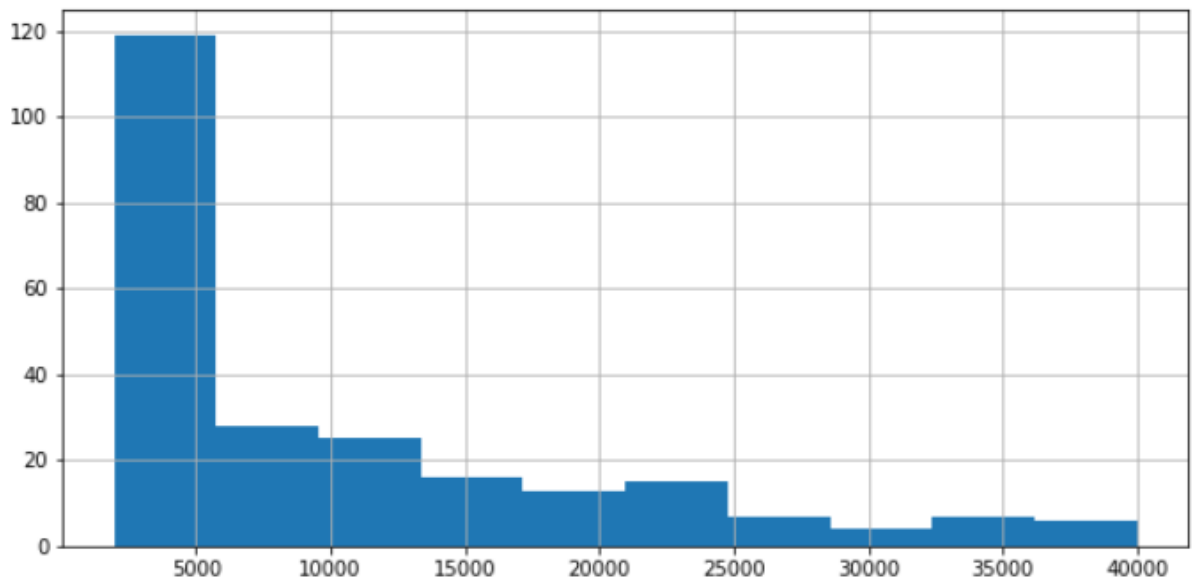
Mão na Massa !

```
[44] # Vamos explorar todas as variaveis da base de dados
     sns.pairplot(Base_Dados);
```

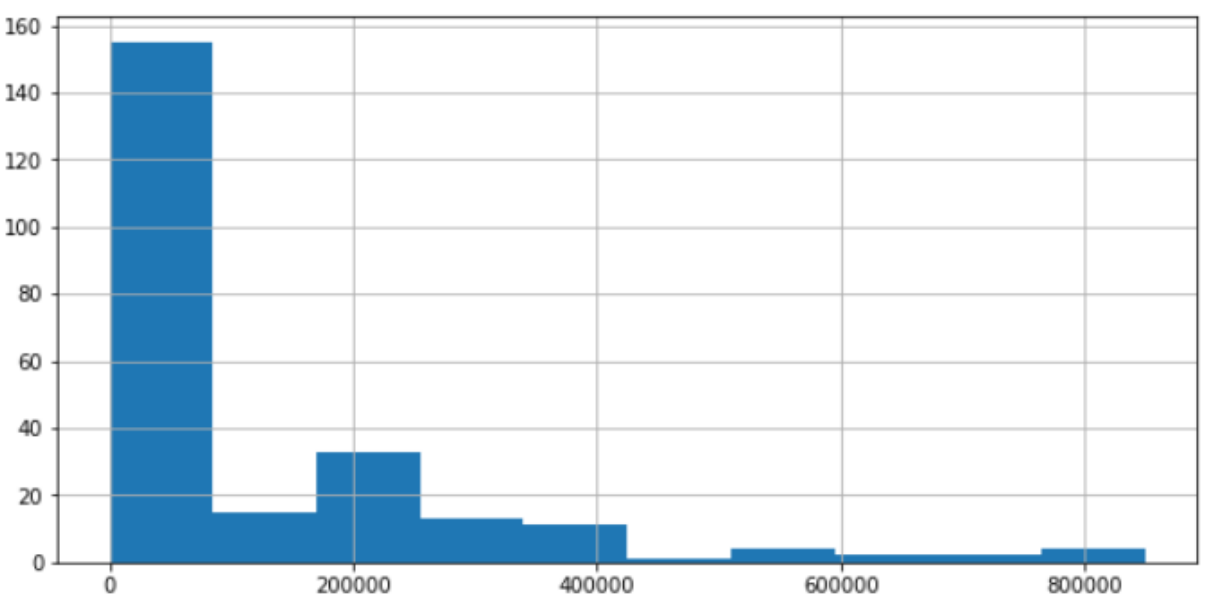


Mão na Massa !

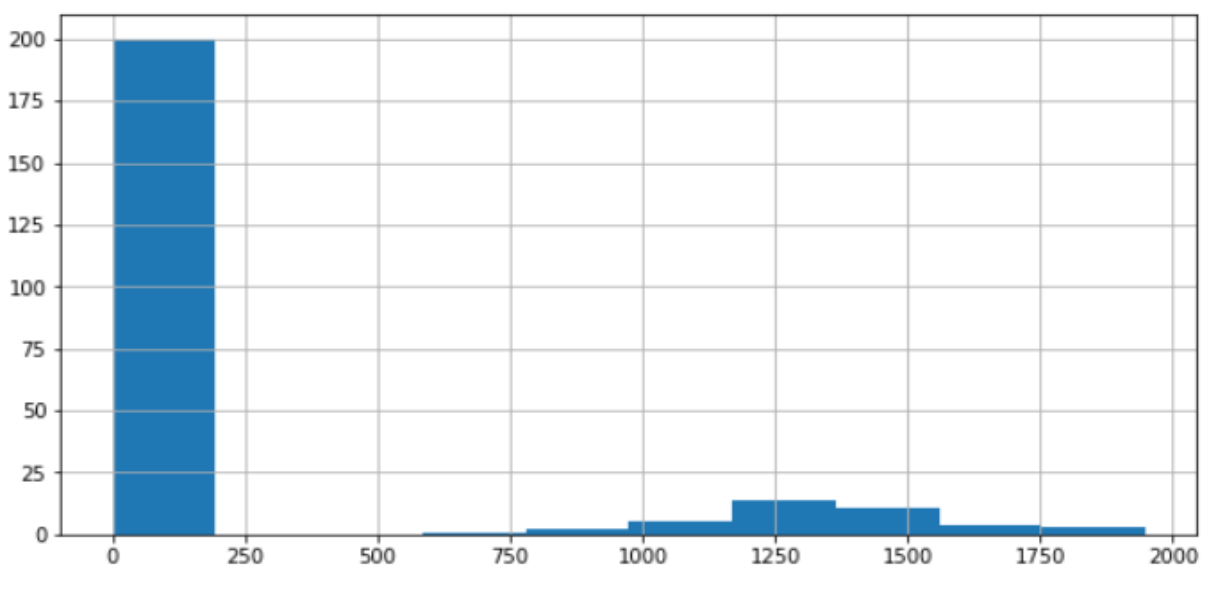
```
[31] # Plotagem da Temperatura
     plt.figure( figsize=(10,5) )
     Base_Dados['Temperature'].hist();
```




```
[32] # Plotagem da Luminosidade das Estrelas
     plt.figure( figsize=(10,5) )
     Base_Dados['L'].hist();
```

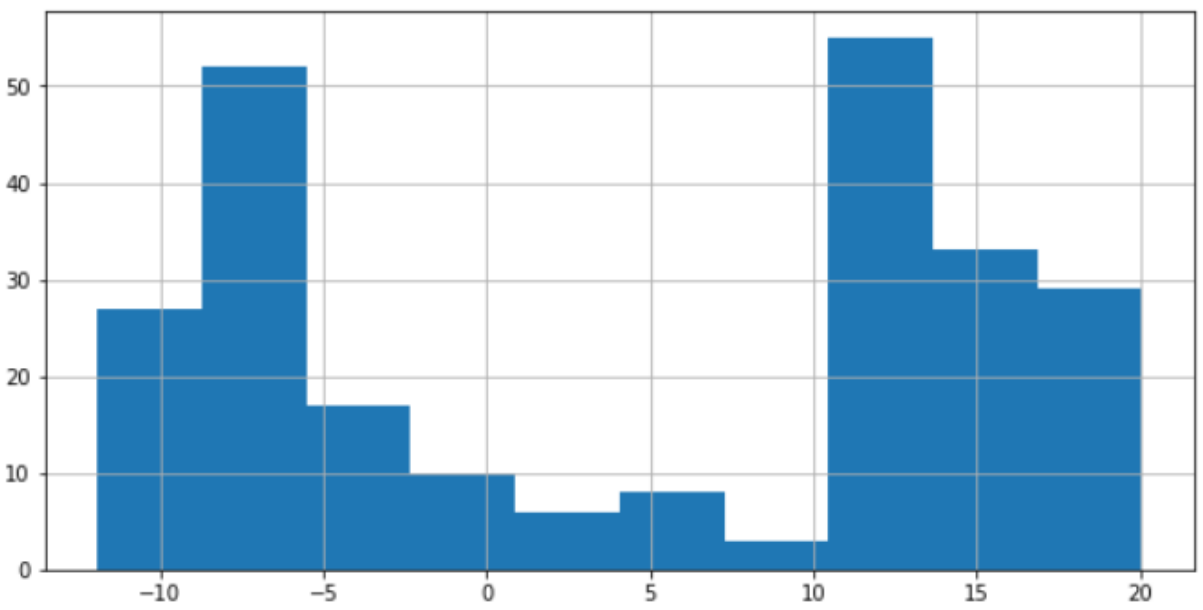



```
     # Plotagem do Tamanho das Estrelas
     plt.figure( figsize=(10,5) )
     Base_Dados['R'].hist();
```

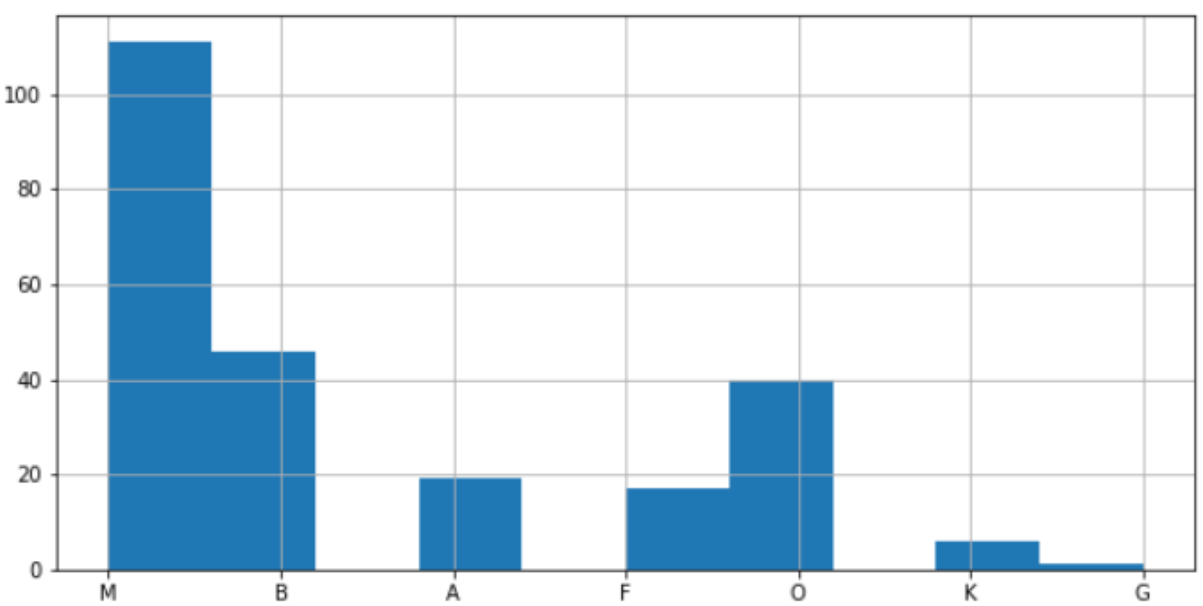


Mão na Massa !

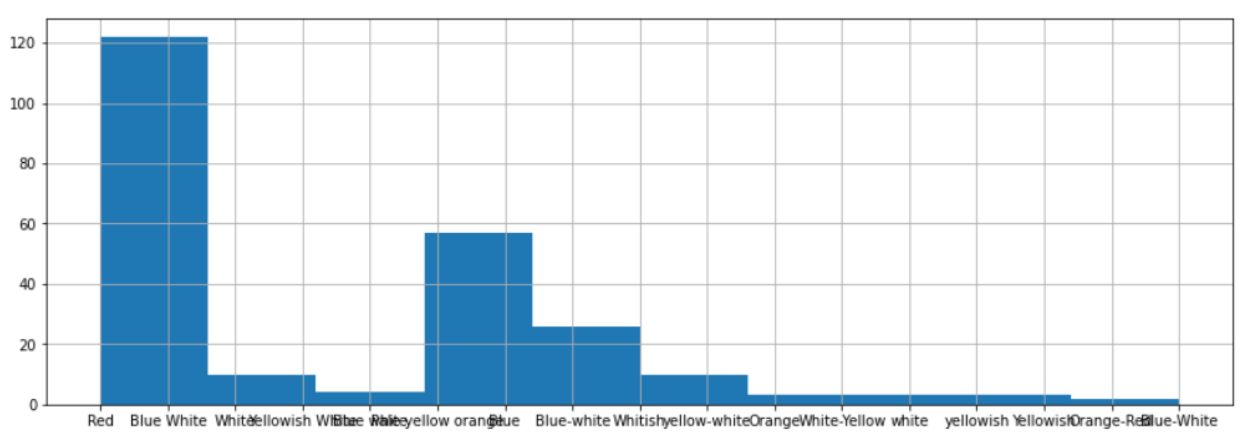
```
[35] # Plotagem da Magnitude das Estrelas
     plt.figure( figsize=(10,5) )
     Base_Dados['A_M'].hist();
```




```
[36] # Plotagem do Tamanho das Estrelas
     plt.figure( figsize=(10,5) )
     Base_Dados['Spectral_Class'].hist();
```

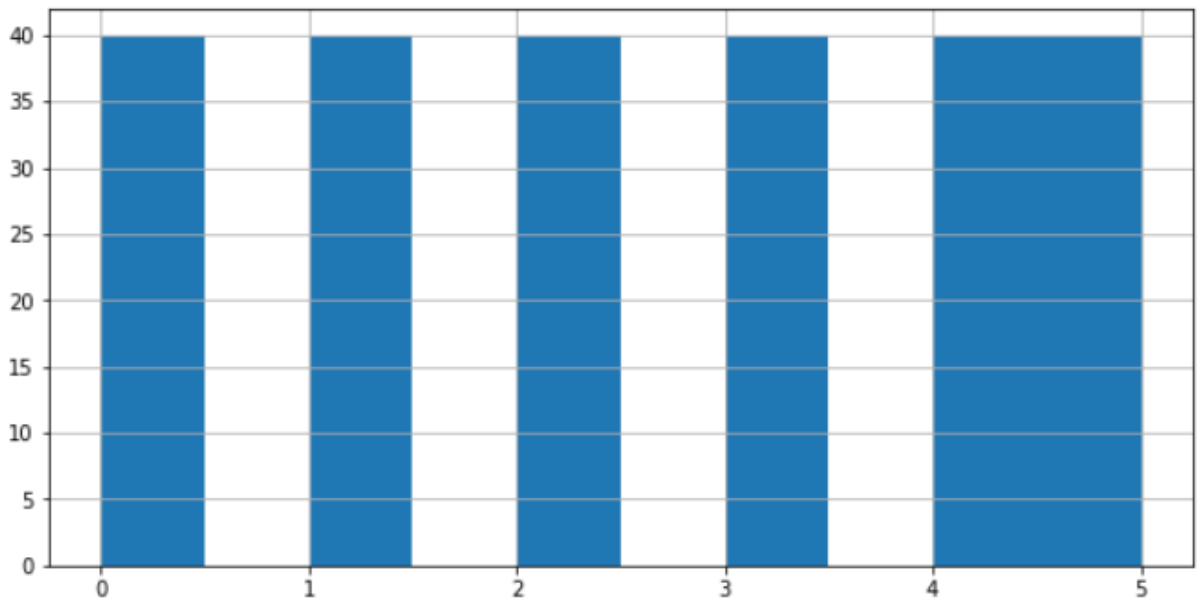



```
[38] # Plotagem as cores das estrelas
     plt.figure( figsize=(15,5) )
     Base_Dados['Color'].hist();
```

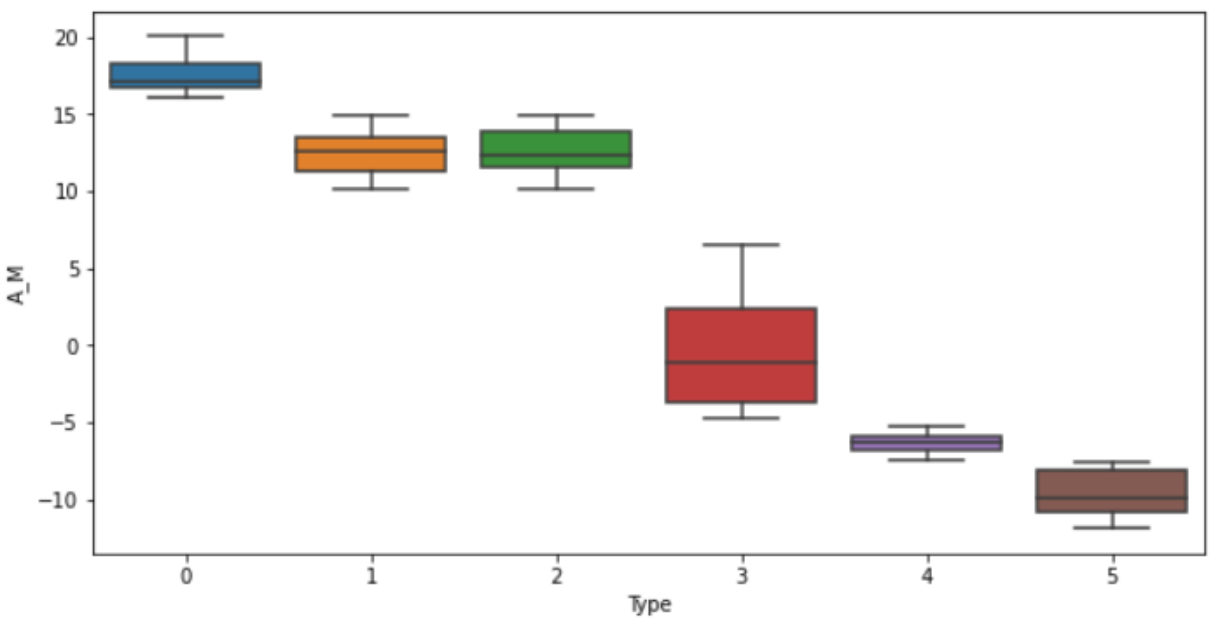


Mão na Massa !

```
[40] # Plotagem o tipo
     plt.figure( figsize=(10,5) )
     Base_Dados['Type'].hist();
```




```
[46] # Plotagem a Magnitude  das Estrelas por Tipo
     plt.figure( figsize=(10,5) )
     sns.boxplot(x='Type', y='A_M', data=Base_Dados);
```



Mão na Massa !

```
[71] # Preparando os dados para o modelo

     # Separando os  Dados de Caracteristicas
     x = Base_Dados.iloc[:,0:4].values

     # Separando os Dado de classificação
     y = Base_Dados.iloc[:, 6:7].values

     # Bilbioteca para fazer o escalonamento dos Dados
     from sklearn.preprocessing import StandardScaler

     # Fazendo o escalonamento do  X
     Escala_x = StandardScaler()
     x = Escala_x.fit_transform( x )
```

```
[73] # Criando o modelo de Floresta de Decisão

     # Importando a função para serpara os dados em teste e treino
     from sklearn.model_selection import train_test_split

     # Separando os dados em teste e treino
     x_treinamento, x_teste, y_treinamento, y_teste = train_test_split(
         x, y,
         test_size = 0.20 )

     # Importando a função da Floresta de Decisão
     from sklearn.ensemble import RandomForestClassifier

     # Definindo o algoritmo
     Algoritmo_Floresta_Decisao = RandomForestClassifier( n_estimators=500 )

     # Aplicando o modelo nos dados
     Algoritmo_Floresta_Decisao.fit( x_treinamento, y_treinamento )
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:16: DataConversionWarning:
  app.launch_new_instance()
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=500,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

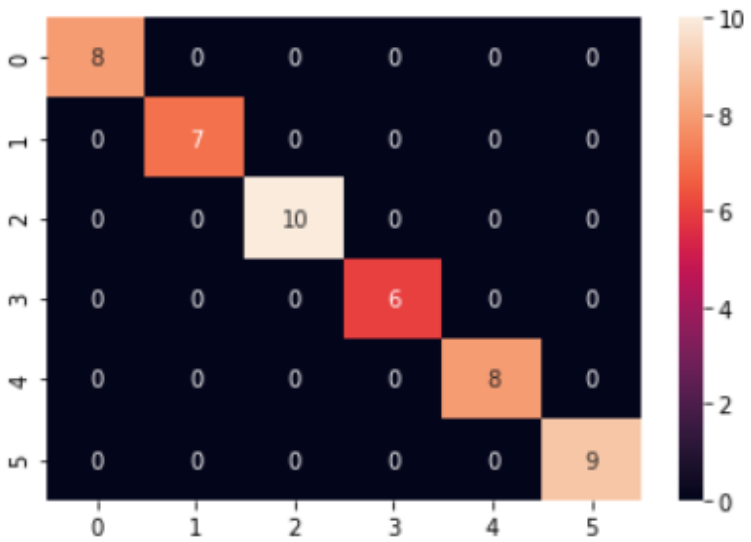
Mão na Massa !

```
[77] # Avaliando o modelo

     # Prevendo os valores do X_teste
     y_predicoes = Algoritmo_Floresta_Decisao.predict( x_teste )

     # Metrica do score de acurracia
     from sklearn.metrics import accuracy_score
     Acuracio_Score = accuracy_score(  y_teste, y_predicoes )
     Acuracio_Score
```

```
1.0
```

```
[79] from sklearn.metrics import confusion_matrix
     Matriz_Confusao = confusion_matrix( y_teste, y_predicoes )
     print( Matriz_Confusao )

     sns.heatmap( Matriz_Confusao, annot=True);
```

```
[[ 8  0  0  0  0  0]
 [ 0  7  0  0  0  0]
 [ 0  0 10  0  0  0]
 [ 0  0  0  6  0  0]
 [ 0  0  0  0  8  0]
 [ 0  0  0  0  0  9]]
```




```
[81] from sklearn.metrics import classification_report
     Previsao = classification_report(y_teste, y_predicoes)
     print( Previsao )
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         7
           2       1.00      1.00      1.00        10
           3       1.00      1.00      1.00         6
           4       1.00      1.00      1.00         8
           5       1.00      1.00      1.00         9

    accuracy                           1.00        48
   macro avg       1.00      1.00      1.00        48
weighted avg       1.00      1.00      1.00        48
```

Mão na Massa !

Minha Conclusão

Nosso modelo apresentou uma acurácia de 100%.

1º Hipótese: Ocorreu um *overfitting* em nosso modelo.

" O modelo overfitting ocorre quando o modelo se adaptou muito bem aos dados com os quais está sendo treinado; porém, não generaliza bem para novos dados. Ou seja, o modelo "decorou" o conjunto de dados de treino, mas não aprendeu de fato o que diferencia aqueles dados para quando precisar enfrentar novos testes. "

Nesse caso seria preciso pedir mais dados para a NASA para podemos testar novamente o modelo e verificamos sua acurácia.

2º Hipótese: Porem a outro ponto, a divisão das estrelas é um **modelo bem definindo** (*2º pagina - tabela das divisão*) o que pode ocorrer é que a **classificação é bem exata** de acordo com as proporções da estrela.

Exemplo: **Não há** como uma estrela anã branca, apresentar uma **dimensão** próximo ou igual a uma gigante vermelha ou vice versa.

Final

Esse guia é um exemplo de uma floresta de decisão.

Guia da documentação caso queira mais detalhes
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Odemir Depieri Jr

Software Engineer Sr
Tech Lead
Specialization AI