Tarefa 1

Autor: Arthur Bridi Guazzelli - RA 234984

Objetivos: leitura, preprocessamento, validação cruzada e regressão linear

In [1]:
```
import pandas as pd
import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

import plotly.graph_objects as go

from sklearn.model_selection import ShuffleSplit, cross_validate
from sklearn.linear_model import LinearRegression
```

1) Leia

- Leia o arquivo solar-flare.csv
- Imprima usando o pandas.head() o início e o fim desse conjunto de dados.

Uma explicação para esses dados esta em https://archive.ics.uci.edu/ml/datasets/Solar+Flare

In [2]:
```
url = 'https://www.ic.unicamp.br/~wainer/cursos/1s2021/432/solar-flare.csv'

data = pd.read_csv(url, sep=' ', header=None, skiprows=1)
data.head(-5)
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	H	A	X	1	3	1	1	1	1	1	0	0	0
1	D	R	O	1	3	1	1	2	1	1	0	0	0
2	C	S	O	1	3	1	1	2	1	1	0	0	0
3	H	R	X	1	2	1	1	1	1	1	0	0	0
4	H	S	X	1	1	1	1	2	1	1	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1056	H	S	X	1	2	1	1	2	1	1	1	0	0
1057	D	S	O	1	2	1	2	2	1	1	0	0	0
1058	H	S	X	1	2	2	2	2	1	1	0	0	0
1059	H	S	X	2	2	1	1	2	1	1	0	0	0
1060	D	R	O	1	3	1	1	2	1	1	0	0	0

1061 rows × 13 columns

2) Converta os atributos categóricos para numéricos

Usando o one-hot-enconder, converta todos os atributos categóricos para numéricos.

Imprima usando o pandas.head() o inicio e o fim desse conjunto de dados transformados.

In [3]:
```
X = pd.get_dummies(data.iloc[:, :-3])
y = data.iloc[:, -3:]

X.head(-5)
```

Out[3]:

	3	4	5	6	7	8	9	0_B	0_C	0_D	…	1_A	1_H	1_K	1_R	1_S	1_X	2_C	2_I	2_O	2_X
0	1	3	1	1	1	1	1	0	0	0	…	1	0	0	0	0	0	0	0	0	1
1	1	3	1	1	2	1	1	0	0	1	…	0	0	0	1	0	0	0	0	1	0
2	1	3	1	1	2	1	1	0	1	0	…	0	0	0	0	1	0	0	0	1	0
3	1	2	1	1	1	1	1	0	0	0	…	0	0	0	1	0	0	0	0	0	1
4	1	1	1	1	2	1	1	0	0	0	…	0	0	0	0	1	0	0	0	0	1
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1056	1	2	1	1	2	1	1	0	0	0	…	0	0	0	0	1	0	0	0	0	1
1057	1	2	1	2	2	1	1	0	0	1	…	0	0	0	0	1	0	0	0	1	0
1058	1	2	2	2	2	1	1	0	0	0	…	0	0	0	0	1	0	0	0	0	1
1059	2	2	1	1	2	1	1	0	0	0	…	0	0	0	0	1	0	0	0	0	1
1060	1	3	1	1	2	1	1	0	0	1	…	0	0	0	1	0	0	0	0	1	0

1061 rows × 23 columns

3) Centering and scaling

Faça o centering and standard scaling para todos os atributos de entrada (convertidos para numéricos)

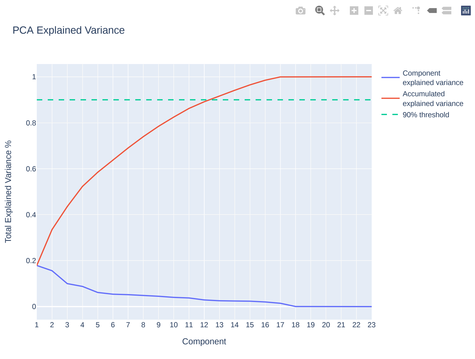
In [4]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

4) PCA

Reduza a dimensionalidade dos atributos de entrada usando PCA. Quantas dimensões restarão se mantivermos 90% da variância dos dados?

Pelo scree plot abaixo, para mantermos 90% da variância dos dados devemos manter 13 dimensões.

In [5]:
```
pca = PCA()
components = pca.fit(X_scaled)
components.explained_variance_ratio_

x = [i+1 for i in range(components.n_components_)]


fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=components.explained_variance_ratio_,
                    mode='lines',
                    name='Component<br>explained variance',
                    ))

acc = [sum(components.explained_variance_ratio_[:i+1]) for i in range(components.n_components_)]
fig.add_trace(go.Scatter(x=x, y=acc,
                    mode='lines',
                    name='Accumulated<br>explained variance',
                    ))

fig.add_trace(go.Scatter(x=x, y=[0.9 for _ in range(len(x))],
                    mode='lines',
                    name='90% threshold',
                    line=dict(dash='dash')))

fig.update(layout=dict(
    title='PCA Explained Variance',
    yaxis=dict(tickmode='linear', title='Component'),
    yaxis=dict(title='Total Explained Variance %'),
    width=800,
    height=600
    )
)
fig.show()
```



In [6]:
```
pca = PCA(n_components=13)
X_reduced = pca.fit_transform(X_scaled)
```

5) Validação cruzada e regressão linear

- Fazendo 5 repetições de uma validação cruzada aleatória com split de 70/30 (70% treino 30% teste).
- Treine 3 regressões lineares, uma para cada um dos 3 atributos de saída.
- Treine no conjunto de treino e meça o RMSE e o MAE deste modelo treinado no conjunto de teste correspondente.
- Imprima o RMSE e o MAE no conjunto de testes de cada uma das 5 repetições. Imprima também a média do RMSE e do MAE.

In [7]:
```
metrics = ['neg_root_mean_squared_error', 'neg_mean_absolute_error']
n_splits = 5

splitter = ShuffleSplit(n_splits=n_splits, test_size=0.3)

cv_results = cross_validate(LinearRegression(), X_reduced, y, cv=splitter, scoring=metrics)

for i in range(n_splits):
    print(f'Split {i+1}: ')
    for name, metric in zip(['RMSE', 'MAE'], metrics):
        print(f'  {name}: {-cv_results["test_" + metric][i]:.4f}')
    print()

for name, metric in zip(['RMSE', 'MAE'], metrics):
    print(f'Mean {name}: {np.mean(-cv_results["test_" + metric]):.4f}')
```

```
Split 1:
  RMSE: 0.4024
  MAE: 0.1885

Split 2:
  RMSE: 0.4515
  MAE: 0.1818

Split 3:
  RMSE: 0.5114
  MAE: 0.2269

Split 4:
  RMSE: 0.3205
  MAE: 0.1574

Split 5:
  RMSE: 0.2947
  MAE: 0.1561

Mean RMSE: 0.3961
Mean MAE: 0.1821
```